P6066

Linguaggio Assembler

Edizione preliminare

olivetti

GR Code 3978440 Z (0)

PREFAZIONE

Questo manuale è diretto ai programmatori che intendono creare moduli scritti nel linguaggio Assembler P6066 da richiamare nei programmi BASIC.

Deve essere utilizzato come manuale di riferimento per capire quali istruzioni sono utilizzabili, come si possono codificare e quali sono le operazioni da esse comandate.

Per i principianti nell'uso di un linguaggio Assembler si consiglia di consultare qualche opera di divulgazione ed introduzione a tale linguaggio.

Riferimenti: Nessuno

Distribuzione: Generale (G)

Edizione Preliminare: Marzo 1980

PUBBLICAZIONE EMESSA DA:

Ing. C. Olivetti & C., S.p.A.
Direzione Marketing Centrale
Servizio Documentazione
77, Via Jervis - 10015 IVREA (Italy)

© 1980, by Olivetti

3978440 Z

INDICE

INTRODUZIONE vii

Linguaggi ad alto livello vii

Linguaggio macchina vii

Linguaggio Assembler vii

1. LA MACCHINA 1-1

Il bit ed il byte 1-1

La semiparola e la parola 1-2

La notazione binaria 1-2

La notazione esadecimale 1-3

La rappresentazione dei numeri 1-4

ZRM 1-9

Lo stack e l'area di overlay 1-9

Registri generali 1-10

Il codice condizione 1-10

L'indirizzamento 1-10

Formato delle istruzioni macchina 1-13

I tipi di istruzioni macchina 1-13

2. L'ASSEMBLER 2-1

Formato delle istruzioni Assembler 2-1

Campo nome 2-1
Campo operazione 2-1
Campo operandi 2-1
Campo commento 2-2

Caratteri accettati dall'Assemblatore 2-2

Tipi di istruzioni Assembler 2-3

Istruzioni esecutive 2-3

Istruzioni direttive per l'Assemblatore 2-8

Macro-istruzioni 2-9

Struttura del linguaggio Assembler 2-9

Gli elementi di una frase Assembler 2-10

Simboli 2-10

Termini autodefiniti 2-11

Riferimento al Location Counter 2-12

Riferimento all'attributo lunghezza dei simboli 2-13

Espressioni 2-14

Literal 2-16

Il programma Assembler 2-17

3. LE ISTRUZIONI ASSEMBLER 3-1

Descrizione delle istruzioni Assembler 3-1

Istruzioni direttive per l'Assemblatore 3-1

CSECT	CONTROL SECTION	3-3
DC	DEFINE CONSTANT	3-5
DROP	DROP	3-13
DS	DEFINE STORAGE	3-15
DSECT	DUMMY SECTION	3-19
EJECT	EJECT	3-21
END	END	3-23
EQU	EQUAL	3-25
ORG	ORGANIZE	3-27
PRINT	PRINT	3-29
SPACE	SPACE	3-31
START	START	3-33
TITLE	TITLE	3-35
USING	USING	3-37
PROC		3-41
PRRET		3-43
A	ADD	3-45
AH	ADD HALFWORD	3-47
AL	ADD LOGICAL	3-49
ALM	ADD LOGICAL MEMORY	3-51
ALR	ADD LOGICAL REGISTER	3-53
ALRI	ADD LOGICAL REGISTER IMMEDIATE	3-55
AM	ADD MEMORY	3-57
AR	ADD REGISTER	3-59
ASA	ALLOCATE STACK AREA	3-61
BAL	BRANCH AND LINK	3-63
BALR	BRANCH AND LINK REGISTER	3-65
BC	BRANCH ON CONDITION	3-67
BCR	BRANCH ON CONDITION REGISTER	3-69
BCT	BRANCH ON COUNT	3-71
BCTR	BRANCH ON COUNT REGISTER	3-73
BXH	BRANCH ON INDEX HIGH	3-75
BXLE	BRANCH ON INDEX LOW OR EQUAL	3-77
C	COMPARE	3-79
CALL	SUBROUTINE CALL	3-81
CBS	BINARY TO ISO CONVERSION	3-83
CDS	DATA TO ISO CONVERSION	3-85
CDM	DATA TO MEMORY CONVERSION	3-87
CH	COMPARE HALFWORD	3-89
CL	COMPARE LOGICAL	3-91

CLC	COMPARE LOGICAL MEMORY	3-93
CLI	COMPARE LOGICAL IMMEDIATE	3-95
CLM	COMPARE LOGICAL MEMORY	3-97
CLR	COMPARE LOGICAL REGISTER	3-99
CM	COMPARE MEMORY	3-101
CMSS	MEMORY TO ISO CONVERSION	3-103
CMST	MEMORY TO STANDARD ISO CONVERSION	3-105
CR	COMPARE REGISTER	3-107
CSBH	ISO TO BINARY CONVERSION	3-109
CSD	ISO TO DATA CONVERSION	3-111
CUMS	MEMORY TO ISO CONVERSION	3-113
DIS	DICOTOMIC SEARCH	3-115
DM	DIVIDE MEMORY	3-117
EX	EXECUTE	3-119
FSA	FREE STACK AREA	3-121
GSA	GET STACK ARGUMENTS	3-123
IC	INSERT CHARACTER	3-125
IM	IMMEDIATE IN MEMORY	3-127
ISO	ISO TEST	3-129
L	LOAD	3-131
LA	LOAD ADDRESS	3-133
LC	LOAD COMPLEMENT	3-135
LCR	LOAD COMPLEMENT REGISTER	3-137
LH	LOAD HALFWORD	3-139
LIE	LOOK FOR IMMEDIATE EQUAL	3-141
LINE	LOOK FOR IMMEDIATE NOT EQUAL	3-143
LM	LOAD MULTIPLE	3-145
LN	LOAD NEGATIVE	3-147
LNR	LOAD NEGATIVE REGISTER	3-149
LPR	LOAD POSITIVE REGISTER	3-151
LR	LOAD REGISTER	3-153
LT	LOAD AND TEST	3-155
LTR	LOAD AND TEST REGISTER	3-157
MLH	MULTIPLY LOGICAL HALFWORD	3-159
MLR	MULTIPLY LOGICAL REGISTER	3-161
MVC	MOVE CHARACTER	3-163
MVCR	MOVE CHARACTER ON REGISTER	3-165
MVI	MOVE IMMEDIATE	3-167
MVN	MOVE NUMERICS	3-169
MVO	MOVE WITH OFFSET	3-171
MVZ	MOVE ZONES	3-173

N	AND	3-175
NC	AND CHARACTER	3-177
NI	AND IMMEDIATE	3-179
NOP	NO OPERATION	3-181
NOPR	NO OPERATION	3-183
NR	AND REGISTER	3-185
O	OR	3-187
OC	OR CHARACTER	3-189
OI	OR IMMEDIATE	3-191
OR	OR REGISTER	3-193
RETS	SUBROUTINE RETURN	3-195
S	SUBTRATC	3-197
SES	SEQUENTIAL SEARCH	3-199
SESM	SEQUENTIAL SEARCH WITH MASK	3-201
SH	SUBTRACT HALFWORD	3-203
SL	SUBTRACT LOGICAL	3-205
SLA	SHIFT LEFT ALGEBRAIC	3-207
SLL	SHIFT LEFT LOGICAL	3-209
SLM	SUBTRACT LOGICAL MEMORY	3-211
SRL	SUBTRACT LOGICAL REGISTER	3-213
SLRI	SUBTRACT LOGICAL REG. IMMEDIATE	3-215
SM	SUBTRACT MEMORY	3-217
SR	SUBTRACT REGISTER	3-219
SRA	SHIFT RIGHT ALGEBRAIC	3-221
SRL	SHIFT RIGHT LOGICAL	3-223
ST	STORE	3-225
STC	STORE CHARACTER	3-227
STH	STORE HALFWORD	3-229
STM	STORE MULTIPLE	3-231
TM	TEST UNDER MASK	3-233
TR	TRANSLATE	3-235
TRT	TRANSLATE AND TEST	3-237
X	EXCLUSIVE OR	3-239
XC	EXCLUSIVE OR CHARACTER	3-241
XI	EXCLUSIVE OR IMMEDIATE	3-243
XR	EXCLUSIVE OR REGISTER	3-245

INTRODUZIONE

Linguaggi ad alto livello

I linguaggi ad alto livello che vengono di solito resi disponibili su ogni calcolatore permettono all'utente di programmare la macchina in modo piuttosto agevole, pur ignorandone la struttura interna.

Se ciò da un lato offre il vantaggio di una facile presa di contatto con la macchina, dall'altro non permette, in talune applicazioni, di utilizzare al meglio le risorse disponibili. Può in questi casi risultare più conveniente programmare la macchina nel suo linguaggio interno che, controllandone direttamente tutti i meccanismi, ne permette un più efficiente sfruttamento.

Linguaggio macchina

Operare direttamente nel linguaggio interno della macchina è d'altro canto piuttosto complicato: è infatti necessario, per ciascuna istruzione, specificare il valore del codice operativo e gli indirizzi assoluti di tutti gli operandi interessati. Tutto ciò è molto disagevole soprattutto nella fase di messa a punto o "debugging" di un programma. In tale momento, infatti, risulta necessario modificare frequentemente il programma inserendo o sopprimendo istruzioni: si dovrebbe allora ricalcolare ogni volta tutti i riferimenti all'interno del programma.

Linguaggio Assembler

In generale il linguaggio Assembler di una macchina è il linguaggio simbolico più vicino al linguaggio interno della macchina stessa. La corrispondenza esistente per le istruzioni dei due linguaggi è di solito diretta. E' quindi utile conoscere la struttura ed i principi di funzionamento e di rappresentazione dei dati di macchina per poterla programmare nel suo linguaggio Assembler.

Un programma scritto in linguaggio Assembler non può peraltro essere eseguito direttamente dalla macchina senza una previa fase di traduzione. Questo processo,

chiamato Assemblaggio, viene svolto da un programma, chiamato Assemblatore, che fa parte del sistema operativo della macchina in questione.

I vantaggi di poter operare con un linguaggio simbolico anzichè con un linguaggio assoluto (quale è il linguaggio interno di ogni macchina) possono essere brevemente riassunti in:

- uso di codici mnemonici in luogo dei corrispondenti codici operativi

- uso di nomi simbolici per l'indirizzamento del programma.

Queste due caratteristiche rendono più agevoli la stesura ed il debugging di un programma. In particolare, essendo l'indirizzamento di tipo simbolico (riferimento a nomi associati alle istruzioni piuttosto che ai loro indirizzi), l'inserimento o la cancellazione di istruzioni non comportano la necessità di ricomputare i riferimenti all'interno del programma. E' infatti l'Assemblatore che, nel processo di traduzione degli indirizzi da simbolici ad assoluti, provvede automaticamente a calcolare gli indirizzi corretti.

1. LA MACCHINA

Rappresentazione dei dati

Il bit ed il byte

L'unità elementare per la rappresentazione in macchina delle informazioni è il "bit": il bit non è indirizzabile od operabile direttamente.

I "bit" sono infatti raggruppati in gruppi di 8 a formare il "byte", e sono i byte gli elementi operabili individualmente.

Il "bit" è una variabile che può assumere uno fra due possibili valori: zero o uno.

Una sequenza di 8 bit permette di rappresentare $2^8 = 256$ diverse configurazioni o "valori" del byte. Infatti, poichè 1 bit può assumere due diversi valori, con un numero n di bit si può ottenere il doppio delle configurazioni possibili con n-1 bit premettendo una volta zero e una volta uno a tutte le configurazioni possibili con n-1 bit. Tali configurazioni sono ottenibili, ad esempio, partendo dalla configurazione di tutti zeri e variando via via tutti i bit, in tutti i modi possibili, partendo da quelli di destra o "meno pesanti". Avremo così:

00000000
00000001
00000010
00000011
00000100
00000101
00000110
00000111
00001000

e così via fino a giungere alla configurazione di tutti uno (11111111).

I byte sono ordinati e numerati consecutivamente in

memoria da sinistra verso destra a partire da zero.

La semiparola e la parola

I byte possono a loro volta essere ulteriormente raggruppati a due o a quattro a formare semiparole o parole.

Una <u>semiparola</u> è un gruppo di 2 byte in cui il byte ad indirizzo più basso (di sinistra) ha un indirizzo multiplo di due. Così 10, 124, 1484 sono indirizzi corretti di semiparole mentre 13, 151, 1937 non lo sono.

Una <u>parola</u> è un gruppo di 4 byte il cui primo byte (di sinistra) abbia un indirizzo multiplo di quattro. Così 20 e 5208 sono indirizzi corretti di parole mentre 10 e 167 sono indirizzi non corretti. Con le opportune istruzioni è possibile, riferendo il primo byte del gruppo, operare contemporaneamente su tutti i byte componenti la semiparola o la parola.

La notazione binaria

La notazione binaria è una convenzione che permette di descrivere le configurazioni di un campo di memoria, indicando il valore di ciascun bit mediante una cifra binaria (zero oppure uno).

Il valore decimale di una determinata configurazione di un byte equivale alla somma di tutti i prodotti:

$$b_i * 2^{i-1}$$

dove $i(\geqslant 1)$ indica la posizione di un bit (crescente da destra verso sinistra), mentre b_i rappresenta il suo valore (Ø o 1).

Ad esempio, la seguente configurazione di bit:

Ø	Ø	Ø	1	Ø	Ø	1	1	b_i
8	7	6	5	4	3	2	1	i

vale 19. Infatti:

$$0*2^{(8-1)} + 0*2^{(7-1)} + 0*2^{(6-1)} + 1*2^{(5-1)} + 0*2^{(4-1)} +$$
$$+ 0*2^{(3-1)} + 1*2^{(2-1)} + 1*2^{(1-1)} = 1*2^4 + 1*2 + 1 =$$
$$= 16 + 2 + 1 = 19$$

La notazione esadecimale

La notazione binaria, descritta al paragrafo precedente, è piuttosto disagevole da utilizzare: si preferisce così considerare il byte come diviso in due semi-byte ed associare ad ogni semi-byte il valore ricavato dalla seguente tabella in cui ognuna delle 16 combinazioni possibili di 4 bit è rappresentata da un simbolo diverso:

Valore Binario	Valore Esadecimale	Valore Decimale
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

E' pertanto possibile rappresentare la configurazione di un byte per mezzo di 2 sole cifre Esadecimali anzichè otto cifre binarie.

Il byte [Ø|Ø|Ø|1|Ø|Ø|1|1] sarà pertanto sinteticamente rappresentato secondo la notazione esadecimale con [1|3] .

E' possibile calcolare il valore di un campo partendo dalla sua notazione Esadecimale in modo del tutto analogo a quanto fatto per la notazione binaria. Così la notazione esadecimale [1|A|3|F] vale [6|7|1|9] nella consueta notazione decimale. Infatti usando questa volta le potenze del 16, avremo:

$1x16^3 + Ax16^2 + 3x16^1 + Fx16^0$

che vale

$1x16^3 + 10x16^2 + 3x16^1 + 15x16^0$
ed infine

$1x4096 + 10x256 + 3x16 + 15x1 = 6.719$

La rappresentazione dei numeri

I dati numerici possono essere rappresentati in uno dei seguenti formati:

- Sorgente (o ISO)
- Binario
- DATA
- Memoria

Formato Sorgente (o ISO): Ogni carattere stampabile è codificato secondo la tabella ISO, mediante una specifica configurazione di un byte.

Le cifre decimali sono rappresentate con le seguenti notazioni esadecimali:

Cifra decimale	Valore del byte in esadecimale
0	30
1	31
2	32
3	33
4	34
5	35
6	36
7	37
8	38
9	39

Il primo semibyte (campo zona) vale sempre '3' mentre il secondo semibyte (campo numerico) ha lo stesso valore della cifra rappresentata.

Un numero in formato sorgente viene rappresentato secondo la seguente sintassi:

$$\left\{\begin{matrix}+\\ \not{b}\end{matrix}\right\} n_1 n_2 \ldots n_x [. n_{x+1} \ldots n_y] [E [\pm] n_{y+1} [n_{y+2}]]$$

Ogni cifra viene rappresentata con un byte, il cui valore è il valore della cifra stessa nella tabella ISO.

Il formato sorgente non utilizza al meglio la memoria disponibile, poichè usufruisce solamente di 10 combinazioni sulle 256 disponibili; usa inoltre fino a 4 byte per rappresentare segno, punto decimale ed esponente.

Così, per esempio, il numero 1237 è rappresentato, in notazione esadecimale, come:

31	32	33	37
byte 1	byte 2	byte 3	byte 4

Il segno, il punto decimale e l'esponente sono indicati con il corrispondente carattere ISO e cioè:

Carattere	Valore del byte
+	2B
-	2D
.	2E
E	45

Esempio:

Il numero +31.7 verrà rappresentato su 5 byte nel seguente modo:

2B	33	31	2E	37
byte 1	byte 2	byte 3	byte 4	byte 5

Questo tipo di rappresentazione non è operabile direttamente con istruzioni della macchina Assembler. A tale scopo è necessario operare una conversione in formato binario.

Formato Binario: Il formato binario prevede due diverse rappresentazioni del numero a seconda che esso sia sempre positivo o che possa assumere valori negativi. Se il numero è sempre positivo, tutti i bit del campo sono dedicati alla rappresentazione del numero stesso. Con un byte potremo così rappresentare numeri fino a 255 (2^8 = 256 configurazioni da Ø a 255), con due byte potremo rappresentare numeri fino a 65535 (2^{16}=65536 configurazioni da Ø a 65535) e così via.

Le istruzioni che operano su tali rappresentazioni sono dette logiche (in quanto non tengono conto dell'aritmetica del segno). Avremo così istruzioni quali ALM o SLM che servono per sommare o sottrarre logicamente fra loro tali campi.

Se il numero può assumere valori negativi il primo bit del campo numerico (il primo a sinistra o di ordine più alto) indica il segno del numero secondo la conversione:

0 per i numeri positivi
1 per i numeri negativi

I restanti bit del campo sono destinati a rappresentare il valore numerico. Nel caso di numeri positivi il valore è rappresentato normalmente; nel caso di numeri negativi i rimanenti bit non indicano il valore del numero ma la sua rappresentazione in complemento a due.

Il complemento a due di un numero binario viene ottenuto sommando 1 al suo complemento ad uno. Il complemento ad uno viene ottenuto sostituendo ad ogni cifra binaria la sua reciproca (cioè zero ad ogni uno ed uno ad ogni zero).

Esempio:

numero	0010	0000	1001	1001
complemento a 1	1101	1111	0110	0110
complemento a 2	1101	1111	0110	0111

Poichè il primo bit è sempre destinato al segno, il massimo ed il minimo numero esprimibili sono:

mezza parola (2 byte) $-32768 = -2^{15} \leq x \leq 2^{15} - 1 = 32767$

parola (4 byte) $-2147483648 = -2^{31} \leq x \leq 2^{31} - 1 = 2147483647$

Le istruzioni che operano su questi campi sono istruzioni di operazioni algebriche come A o S.

Nelle operazioni di questo tipo si possono verificare situazioni di "overflow" o "underflow", quando i riporti del segno e del bit più pesante sono discordanti.

Ad esempio (useremo per semplicità campi di 4 bit):

11ØØ	[-4]
1ØØ1	[-7]

Risultato della somma:

Ø1Ø1	[+5]

Riporto del segno: 1
Riporto del bit più pesante: Ø

Il Formato DATA: La rappresentazione di un numero in formato DATA è il risultato di una conversione di un numero dal formato sorgente. Il formato DATA ha la seguente struttura:

A	B	ESP_1	ESP_2	S	N_1		N_{x+1}	N_x
byte 1		byte 2	byte 3	byte 4				byte n

dove:

A = numero di zeri non significativi

B = numero di cifre significative

ESP_1 = rappresenta l'esponente così come era indicato nel formato sorgente

ESP_2 = rappresenta l'esponente del numero in formato normalizzato (in cui, cioè, il punto decimale è subito a destra della prima cifra significativa)

S = segno del numero ('B' per + e 'D' per -)

$N_1 \ldots N_x$ = rappresentano i campi numerici delle x cifre significative del numero (se x è pari, il semibyte di destra dell'ultimo byte contiene un valore non significativo).

Questo formato non è direttamente operabile dalle istruzioni algebriche della macchina Assembler.

Il Formato Memoria: La rappresentazione in formato Memoria è il risultato della conversione di un numero dalla sua rappresentazione in formato DATA.

Si distinguono due formati:

- semplice precisione:

  Il numero è rappresentato su 4 byte.

  I primi tre byte contengono la mantissa del numero, (complessivamente sei cifre), le eventuali cifre oltre la sesta vengono troncate nella conversione.

  Il primo bit di sinistra del quarto byte rappresenta il segno del numero (1='+' 0='-'), i rimanenti sette bit riportano l'esponente del numero. L'esponente è rappresentato in complemento a due più 64 avremo cioè che:

Ø = 4Ø	-1 = 3F
1 = 41	-2 = 3E
.	
.	
.	
63 = 7F	-63 = Ø1

  Il punto decimale è inteso a destra della prima cifra.

- doppia precisione:

  Il numero è rappresentato su 8 byte.

  Il primo semibyte del primo byte riporta il segno del numero (B='+', D='-').

  Seguono, dal secondo semibyte del primo byte fino al secondo semibyte del settimo byte, fino a 13 cifre di mantissa del numero. Le eventuali cifre oltre la tredicesima vengono troncate.

  Sull'ottavo byte è infine riportato, su 8 bit, l'esponente del numero rappresentato in complemento a due + 128 (decimale).

  Il punto decimale è inteso a destra della prima cifra.

Anche questa rappresentazione di dati non è direttamente operabile con istruzioni Assembler.

ZRM

La ZRM (Zona Riservata di Memoria) è una zona di memoria che contiene alcune informazioni circa la configurazione di macchina, lo stato del sistema e i contenuti della'area utente, che sono gestite dal sistema operativo ma cui può accedere anche il programma utente. In quest'area sono contenuti, fra gli altri, anche i descrittori dell'area di overlay, dello stack e dell'area riservata al software di base.

Per una descrizione dei contenuti della ZRM si veda il "Manuale del Sistema".

Lo stack e l'area di overlay

Lo stack è un'area della macchina (ad indirizzi alti di memoria) che viene gestita secondo una politica LIFO (Last In First Out): l'ultimo dato arrivato è il primo ad essere preso in considerazione).

Quest'area è usata da alcune istruzioni macchina; è tuttavia possibile, per il programma, controllarla direttamente per ricavarne aree temporanee di lavoro.

L'area di overlay viene usata per caricare in memoria centrale segmenti di sistema operativo o di programmi residenti su memoria di massa. E' il sistema operativo che si prende cura di caricare in quest'area gli opportuni moduli a fronte dei relativi richiami e, se del caso, di compattarla opportunamente per permettere sia una buona utilizzazione dello spazio di memoria sia una limitazione del numero di accessi al disco per il caricamento dei moduli necessari.

Lo stack e l'area di overlay sono di solito adiacenti e in competizione con la stessa area. L'area di overlay cresce verso indirizzi alti di memoria mentre lo stack viene riempito da indirizzi alti di memoria verso indirizzi via via più bassi.

Se una della due aree cerca di occupare un'area già occupata dall'altra, viene generata una condizione di errore. I delimitatori di queste aree sono contenuti in ZRM.

Per una descrizione più dettagliata si veda il "Manuale del Sistema".

Registri generali

I registri generali, in numero di 16, sono dei campi di memoria capaci di contenere un numero intero positivo o negativo, in formato binario. Le principali funzioni svolte dai registri generali si possono così riassumere:

- costituiscono aree di lavoro per operazioni aritmetiche su dati in formato binario

- svolgono una funzione essenziale per il meccanismo di indirizzamento

- forniscono supporto ad alcune operazioni logiche

Un registro generale è formato da una parola (4 byte). Di conseguenza il numero di diverse configurazioni possibili di un registro è pari a 2^{32}. Dato che i numeri negativi vengono rappresentati in complemento a due, i numeri rappresentabili in una parola (e quindi in un registro) variano fra -2^{31} e $2^{31}-1$ (estremi inclusi).

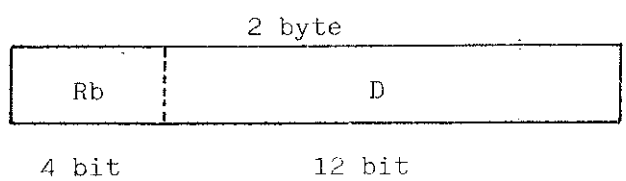
Il codice condizione

Il codice condizione non è un campo esplicitamente operabile dal programmatore.

Ad esso accedono alcune istruzioni di macchina che gli assegnano un valore in funzione delle diverse condizioni verificatesi durante l'esecuzione della istruzione stessa, o che effettuano o meno l'operazione prevista (ad esempio un salto), a seconda dei contenuti di tale campo.

L'indirizzamento

L'indirizzo nelle istruzioni del P6066 è rappresentato su due byte nel seguente formato:



il valore del primo campo di 4 bit indica uno dei sedici registri generali presenti nella macchina (da 0 a F in esadecimale); il valore dei successivi 12 bit (secondo campo) è un numero (rappresentato in binario) che può variare fra Ø e 4095.

L'indirizzo, rappresentato su due byte, è calcolato come somma del contenuto del registro specificato nel primo campo Rb (base) e del contenuto del secondo campo D (displacement o scostamento). In modo simbolico si indicherà che l'indirizzo I vale:

$$I = (Rb) + D$$

dove con la notazione '()' si sta ad indicare "il contenuto di".

Esempi:

Sia in un dato istante il contenuto dei registri generali:

	Valore decimale
Registro 0	2000
Registro 1	16815
.....	.....
Registro 5	27612
.....	.....
Registro 12	18029
.....	.....

Siano altresì presenti in memoria due campi di due byte ciascuno:

	Contenuto				Valore esadecimale			
campo 1	0101	0000	1100	1001	5	Ø	C	9
					Rb	D		
Campo 2	1100	0000	0100	1100	C	Ø	4	C
					Rb	D		

Se questi campi rappresentano indirizzi, allora si riferiscono alle locazioni di memoria 27813 e 18105 rispettivamente.

Infatti nel primo caso il contenuto del registro 5 vale 27612, che va sommato allo scostamento (201); l'indirizzo espresso è pertanto 27813.

Nel secondo caso il contenuto del registro 12 vale 18029 che va sommato allo scostamento (76); l'indirizzo espresso è pertanto 18105.

Un registro generale, quando è usato per esprimere un indirizzo di memoria, è detto registro base.

In alcune istruzioni macchina per specificare un indirizzo si fa riferimento, oltre che al registro base e allo scostamento, ad un altro registro generale che concorre alla determinazione dell'indirizzo. In questo caso l'indirizzo effettivo si ricava come somma dello scostamento del contenuto del registro base e del contenuto di quest'ultimo registro, chiamato registro indice.

Esempio:

scostamento	: 120	
registro base	: 7	(contenuto: 2328)
registro indice	: 11	(contenuto: 10)
indirizzo risultante	: 120 + 2328 + 10 = 2458	

Nota: Il registro Ø non può essere mai indicato come registro base nè come registro indice. Uno zero nel campo "registro base" o nel campo "registro indice" di una istruzione significa che, nel calcolare l'indirizzo, non si tiene conto della componente "registro base" o della componente "registro indice".

Rilocabilità. La rilocabilità è una caratteristica dei programmi che ne permette l'eseguibilità a prescindere dall'indirizzo di caricamento del programma stesso in memoria.

Con uno schema di indirizzamento come quello descritto, un programma è quasi immediatamente rilocabile.

Se tutti gli indirizzi nel programma sono risolti rispetto ad uno stesso registro è infatti sufficiente caricare tale registro con un valore opportuno per rendere immediatamente eseguibile il programma dato, qualunque sia l'indirizzo a cui viene caricato in memoria.

Formato delle istruzioni macchina

I tipi di istruzioni macchina

Le istruzioni del linguaggio macchina hanno lunghezze e formati variabili a seconda del numero e del tipo degli operandi su cui devono operare. Questi diversi formati sono riconducibili a cinque gruppi distinti. Caratteristica comune a tutte le istruzioni è che il primo campo (1 byte), contiene il codice operativo dell'istruzione stessa (ovvero il codice con cui si riconosce il tipo di funzione che deve essere eseguita sugli argomenti specificati di seguito).

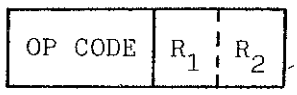
I 5 formati sono:

RR
RX
RS
SI
SS

Per la descrizione dei formati delle istruzioni, si fa riferimento a simboli mnemonici che vengono di seguito riportati e spiegati:

Simbolo	Significato
M	maschera
R	registro generale
D	valore di scostamento
B	registro base
X	registro indice
S	indirizzo implicito (in genere un simbolo)
I	immediato (in genere termine autodefinito)
L	lunghezza
N	numero di argomenti
1,2,3...n	riferimento al 1°,2°...ennesimo operando della fase sorgente

Formato RR. Lunghezza 1 semiparola = 2 byte.



1° byte 2° byte

OP CODE	M_1	R_2
1° byte	2° byte	

OP CODE	I_2	R_1
1° byte	2° byte	

OP CODE	I
1° byte	2° byte

Esempio:

1A	5	C

I valori dei campi sono espressi nella notazione esadecimale:

1A è il codice operativo corrispondente all'operazione di somma fra i contenuti di due registri specificati come operandi (AR).

5 e C indicano i registri suddetti (C è il registro n. 12).

Il risultato della somma è posto nel primo operando, cioè nel registro 5.

Sia il contenuto dei registri prima dell'esecuzione:

registro 5 12.123

registro 12 14.024

dopo l'esecuzione dell'istruzione si avranno i seguenti contenuti:

registro 5 26.147

registro 12 14.024

Formato RX. Lunghezza 2 semiparole = 4 byte.

OP CODE	R_1	X_2	B_2	D_2
1 byte	1 byte		$\frac{1}{2}$ byte	$1\frac{1}{2}$ byte

OP CODE	M_1	X_2	B_2	D_2

OP CODE	X_1	B_1	D_1

Questo formato è usato per istruzioni relative a operazioni fra registri e memoria centrale, come per esempio, il confronto fra il contenuto di un campo di memoria e il contenuto di un registro, oppure la memorizzazione del contenuto di un registro su un campo di memoria.

L'indirizzo di memoria riferito nell'istruzione è ottenuto sommando lo scostamento 'D' al contenuto del registro base 'B' e a quello del registro indice 'X', cioè:

$$I = D + (B) + (X)$$

Esempio:

50	E	2	4	8A4

dove:

. 50 è il codice operativo corrispondente alla operazione di memorizzazione del contenuto di un registro in una parola di memoria (ST)

. E indica il registro generale numero 14 il cui contenuto va trasferito in memoria

. 2 indica il registro indice

. 4 indica il registro base

. 8A4 è lo scostamento (pari a 2212 in decimale)

Gli ultimi tre elementi individuano la parola in memoria in cui trasferire il contenuto del registro.

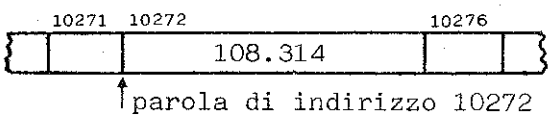
Sia il contenuto dei registri prima dell'esecuzione:

registro 2	40
registro 4	8.020
registro 14	108.314

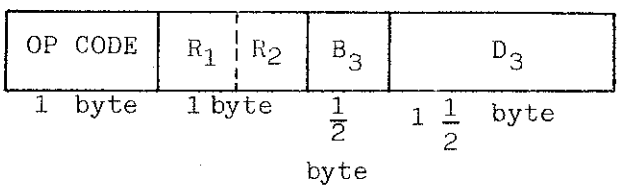
La parola di memoria in cui viene trasferito il contenuto del registro 14 inizia dall'indirizzo:

40+8020+2212=10272

Dopo l'esecuzione dell'istruzione il contenuto di tale parola varrà:



<u>Formato RS</u>. Lunghezza 2 semiparole = 4 byte.



Esempio:

9	Ø	7	9	5	4	7	A

dove:

. 90 è il codice operativo corrispondente all'operazione di trasferimento in memoria del contenuto di più registri consecutivi, individuati dagli estremi R_1, R_2.

. 7 e 9 indicano i registri suddetti: si vuole cioè

trasferire il contenuto dei registri 7, 8 e 9, considerati come un unico campo.

. 5 Indica il registro base.

. 47A E' lo scostamento (pari a 1146 in decimale).

Sia il contenuto dei registri prima dell'esecuzione:

registro 5	48.248							
registro 7	0	0	0	4	3	1	2	7
registro 8	0	0	0	0	0	7	A	2
registro 9	0	0	0	0	A	2	F	1

L'indirizzo del secondo operando sarà:

48248+1146=49394

dopo l'esecuzione dell'istruzione si avrà:

49. 394												
00	04	31	27	00	00	07	A2	00	00	A2	F1	

indirizzo di memoria

Formato SI. Lunghezza 2 semiparole = 4 byte.

OP CODE	I_2	B_1	D_1
1 byte	1 byte	$\frac{1}{2}$ byte	1 $\frac{1}{2}$ byte

Esempio:

9	5	4	1	3	F	2	0

dove:

. 95 è il codice operativo corrispondente all'operazione di comparazione fra una singola po-

sizione di memoria e il dato contenuto in I (CLI).

. 41 rappresenta la lettera A (secondo il codice ISO).

. 3 è il registro base.

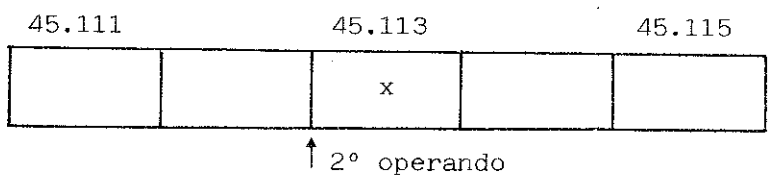
. F20 è lo scostamento (pari a 3840 in decimale).

Sia il contenuto del registro 3 prima dell'esecuzione:

registro 3 | 41.273 |

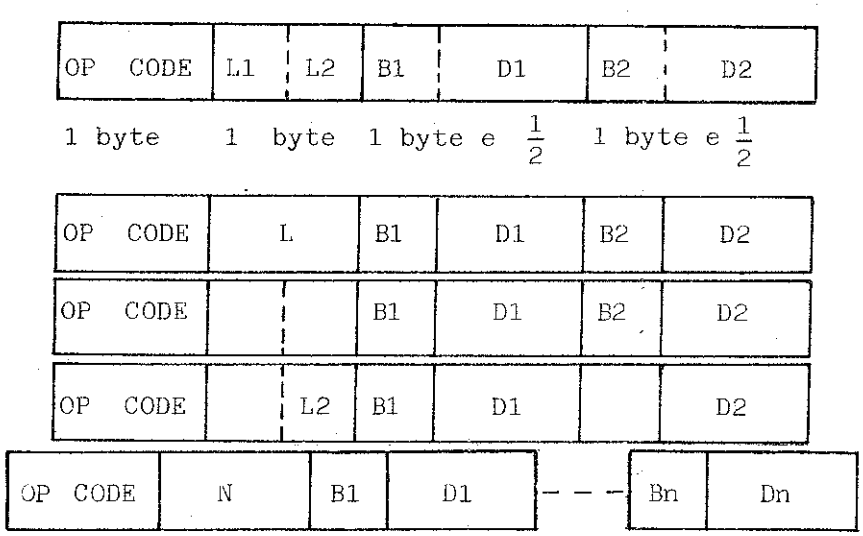
l'indirizzo del secondo operando risulta:

41.273+3.840=45.113



Il contenuto x di tale byte viene confrontato con il carattere A (primo operando) ed al Codice di Condizione viene assegnato un valore diverso a seconda che x risulti uguale, minore o maggiore di A.

<u>Formato SS</u>. Lunghezza 3 o più semiparole = 6 byte o più.



Esempio:

F B	5 3	2	4 A 3	5	8 1 B
byte 1	byte 1	byte 3	byte 4	byte 5	byte 6

dove:

. FB è il codice operativo corrispondente all'operazione di addizione logica fra due campi di memoria (ALM).

. 2 e 4A3 rappresentano l'indirizzo del primo operando.

. 5 e 81B rappresentano l'indirizzo del secondo operando.

. 5 indica che il primo operando è lungo 6 byte.

. 3 indica che il secondo operando è lungo 4 byte.

La lunghezza effettiva di un operando è, di fatto, sempre superiore di 1 al valore indicato nel corpo dell'istruzione, partendo questo valore da zero (lunghezza 1).

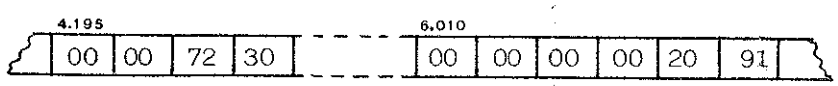
Il risultato della somma viene posto nel primo operando.

Sia il contenuto dei registri prima dell'esecuzione:

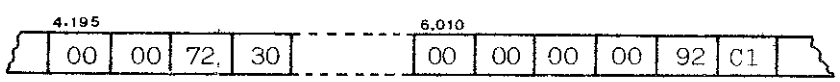
registro 2 4823

registro 5 2120

il primo operando avrà indirizzo pari a 6010 ed il secondo operando indirizzo pari a 4195. Supponendo che essi abbiano i seguenti valori:



dopo l'esecuzione i valori saranno i seguenti:



2. L'ASSEMBLER

Formato delle istruzioni Assembler

L'Assemblatore accetta due diversi tipi di frasi sorgenti:

- Istruzioni Assembler
- Frasi di commento.

Le istruzioni Assembler sono composte dai seguenti 4 campi:

- Nome
- Operazione
- Operandi
- Commento.

Ogni campo deve essere separato dal precedente da uno o più caratteri spazio.

Campo nome

Il campo nome è usato per attribuire un nome simbolico alle frasi sorgenti. Deve sempre iniziare dalla prima posizione della frase ed è costituito al massimo di 8 caratteri alfanumerici, il primo dei quali deve essere alfabetico. Il nome è sempre opzionale tranne che per le istruzioni DSECT ed EQU. Se il primo carattere della frase sorgente è un carattere "spazio", l'Assembler assume tale frase come priva di nome.

Campo operazione

Questo campo deve comparire obbligatoriamente in tutte le istruzioni. Deve riportare il codice simbolico di una delle Istruzioni Assembler. Ha una lunghezza massima di 6 caratteri.

Campo operandi

Questo campo deve contenere la specifica degli operandi previsti dalla istruzione. Tali operandi possono identificare e/o descrivere:

- registri
- aree di memoria
- valori immediati

- maschere
- lunghezza di aree di memoria o tipi di dati
- informazioni per l'Assemblatore.

Campo commento

Questo campo è opzionale e contiene delle note che l'utente può includere nel tabulato emesso dal programma Assemblatore. Nel caso di istruzioni aventi il campo operandi opzionale, quando tale campo è omesso l'eventuale commento deve essere preceduto dalla virgola, a sua volta preceduta e seguita da uno o più caratteri 'spazio'.

Le frasi commento sono identificate dal carattere "*" posto nella prima posizione della frase. Queste frasi non generano alcuna azione da parte dell'Assemblatore, ma vengono semplicemente stampate nel tabulato in output dal programma. Possono contenere qualsiasi tipo di carattere accettato dall'Assemblatore.

Caratteri accettati dall'Assemblatore

Le frasi sorgente Assembler sono scritte utilizzando i seguenti caratteri del codice ISO esteso:

- caratteri alfabetici: da A a Z

- caratteri speciali:

  . delimitatori : virgola [,]
  parentese aperta [(]
  parentesi chiusa [)]

  . spazio : [carattere spazio]

  . operatori : segno più [+]
  segno meno [-]
  segno asterisco [*]
  segno barra [/]

  . caratteri speciali : segno uguale [=]

- caratteri numerici: da Ø a 9.

Oltre ai caratteri citati possono fare parte di una frase sorgente tutti i caratteri del codice ISO esteso, purchè facciano parte di stringhe di dati tipo carattere.

Esempio:

DC C'?'

Tipi di istruzioni Assembler

Le istruzioni del linguaggio Assembler si dividono essenzialmente in 3 tipi di frasi:

- istruzioni esecutive
- direttive per l'Assemblatore
- macro-istruzioni.

Istruzioni esecutive

Sono la rappresentazione simbolica "uno a uno" delle istruzioni macchina. A fronte di ogni frase esecutiva del programma sorgente, l'Assemblatore provvede a generare una istruzione macchina oggetto; sono definiti codici mnemonici per tutte le istruzioni macchina. Le istruzioni esecutive possono essere divise in due gruppi:

- istruzioni standard
- istruzioni speciali.

Le istruzioni standard sono elencate nella seguente tabella raggruppate per gruppi funzionali.

Istruzioni aritmetiche.

MNEMONICO	DESCRIZIONE
A	ADD
AH	ADD HALFWORD
AL	ADD LOGICAL
ALM	ADD LOGICAL MEMORY
ALR	ADD LOGICAL REGISTER
ALRI	ADD LOGICAL REGISTER IMMEDIATE
AM	ADD MEMORY
AR	ADD REGISTER
DM	DIVIDE MEMORY
MLH	MULTIPLY LOGICAL HALFWORD
MLR	MULTIPLY LOGICAL REGISTER
S	SUBTRACT
SH	SUBTRACT HALFWORD
SL	SUBTRACT LOGICAL
SLM	SUBTRACT LOGICAL MEMORY

SLR	SUBTRACT LOGICAL REGISTER
SLRI	SUBTRACT LOGICAL REGISTER IMMEDIATE
SM	SUBTRACT MEMORY
SR	SUBTRACT REGISTER

Istruzioni di confronto.

MNEMONICO	DESCRIZIONE
C	COMPARE
CH	COMPARE HALFWORD
CL	COMPARE LOGICAL
CLC	COMPARE LOGICAL CHARACTER
CLI	COMPARE LOGICAL IMMEDIATE
CLM	COMPARE LOGICAL MEMORY
CLR	COMPARE LOGICAL REGISTER
CM	COMPARE MEMORY
CR	COMPARE REGISTER
ISO	ISO TEST
TM	TEST UNDER MASK

Istruzioni di salto.

MNEMONICO	DESCRIZIONE
B	BRANCH
BAL	BRANCH AND LINK
BALR	BRANCH AND LINK REGISTER
BC	BRANCH ON CONDITION
BCR	BRANCH ON CONDITION REGISTER
BCT	BRANCH ON COUNT
BCTR	BRANCH ON COUNT REGISTER
BE	BRANCH ON EQUAL
BH	BRANCH ON HIGH
BL	BRANCH ON LOW
BM	BRANCH ON MINUS
BNE	BRANCH ON NOT EQUAL
BNH	BRANCH ON NOT HIGH
BNL	BRANCH ON NOT LOW
BNM	BRNACH ON NOT MINUS

BNO	BRANCH IF NOT ONES
BNP	BRANCH ON NOT PLUS
BNZ	BRANCH ON NOT ZERO
BO	BRANCH ON OVERFLOW
BP	BRANCH ON PLUS
BR	BRANCH
BXH	BRANCH ON INDEX HIGH
BXLE	BRANCH ON INDEX LOW OR EQUAL
BZ	BRANCH ON ZERO

Istruzioni Booleane.

MNEMONICO	DESCRIZIONE
N	AND
NC	AND CHARACTER
NI	AND IMMEDIATE
NR	AND REGISTER
O	OR
OC	OR CHARACTER
OI	OR IMMEDIATE
OR	OR REGISTER
X	EXCLUSIVE OR
XC	EXCLUSIVE OR CHARACTER
XI	EXCLUSIVE OR IMMEDIATE
XR	EXCLUSIVE OR REGISTER

Istruzioni di manipolazione registri.

MNEMONICO	DESCRIZIONE
IC	INSERT CHARACTER
L	LOAD
LA	LOAD ADDRESS
LC	LOAD COMPLEMENT
LCR	LOAD COMPLEMENT REGISTER
LH	LOAD HALFWORD
LM	LOAD MULTIPLE
LN	LOAD NEGATIVE
LNR	LOAD NEGATIVE REGISTER

LPR	LOAD POSITIVE REGISTER
LR	LOAD REGISTER
LT	LOAD AND TEST
LTR	LOAD AND TEST REGISTER
SLA	SHIFT LEFT ALGEBRAIC
SLL	SHIFT LEFT LOGICAL
SRA	SHIFT RIGHT ALGEBRAIC
SRL	SHIFT RIGHT LOGICAL
ST	STORE
STC	STORE CHARACTER
STH	STORE HALFWORD
STM	STORE MULTIPLE

Istruzioni di spostamento.

MNEMONICO	DESCRIZIONE
IM	IMMEDIATE IN MEMORY
MVC	MOVE CHARACTER
MVCR	MOVE CHARACTER ON REGISTER
MVI	MOVE IMMEDIATE
MVN	MOVE NUMERICS
MVO	MOVE WITH OFFSET
MVZ	MOVE ZONES

Istruzioni di traduzione.

MNEMONICO	DESCRIZIONE
TR	TRANSLATE
TRT	TRANSLATE AND TEST

Istruzioni particolari.

MNEMONICO	DESCRIZIONE
EX	EXECUTE
NOP	NO OPERATION
NOPR	NO OPERATION REGISTER

Le istruzioni speciali sono suddivise ulteriormente nelle sottoclassi elencate di seguito.

Istruzioni di gestione dello stack.

MNEMONICO	DESCRIZIONE
ASA	ALLOCATE STACK AREA
FSA	FREE STACK AREA
GSA	GET STACK ARGUMENT

Istruzioni di conversione.

MNEMONICO	DESCRIZIONE
CBS	BINARY TO ISO CONVERSION
CDM	DATA TO MEMORY CONVERSION
CDS	DATA TO ISO CONVERSION
CMSS	MEMORY TO ISO CONVERSION
CMST	MEMORY TO STANDARD CONVERSION
CSBH	ISO TO BINARY HALFWORD CONVERSION
CSD	ISO TO DATA CONVERSION
CUMS	MEMORY TO ISO CONVERSION

Istruzioni di ricerca in tabella.

MNEMONICO	DESCRIZIONE
DIS	DICOTOMIC SEARCH
LIE	LOOK FOR IMMEDIATE EQUAL
LINE	LOOK FOR IMMEDIATE NOT EQUAL
SES	SEQUENTIAL SEARCH
SESM	SEQUENTIAL SEARCH WITH MASK

Istruzioni direttive per l'Assemblatore

Le istruzioni direttive per l'Assemblatore servono per dirigere la traduzione del programma e/o per richiedere all'Assemblatore particolari funzioni ausiliarie. Si possono individuare i seguenti sottogruppi di istruzioni:

Direttive di indirizzamento. Servono per controllare l'adozione dei registri come registri base per la traduzione degli indirizzi specificati in modo simbolico:

- USING
- DROP.

Direttive di controllo del programma. Governano la generazione di codice eseguibile o controllano la definizione di simboli nel programma.

- CSECT
- DSECT
- START
- ORG
- END.

Direttive di definizione dati e aree. Servono per definire costanti o riservare aree all'interno di un programma.

- DC
- DS.

Direttive di definizione simboli. Permettono di definire simboli di programma.

- EQU
- EXT.

Direttive di controllo della lista del programma. Permettono di controllare il formato della lista prodotta dall'Assemblatore.

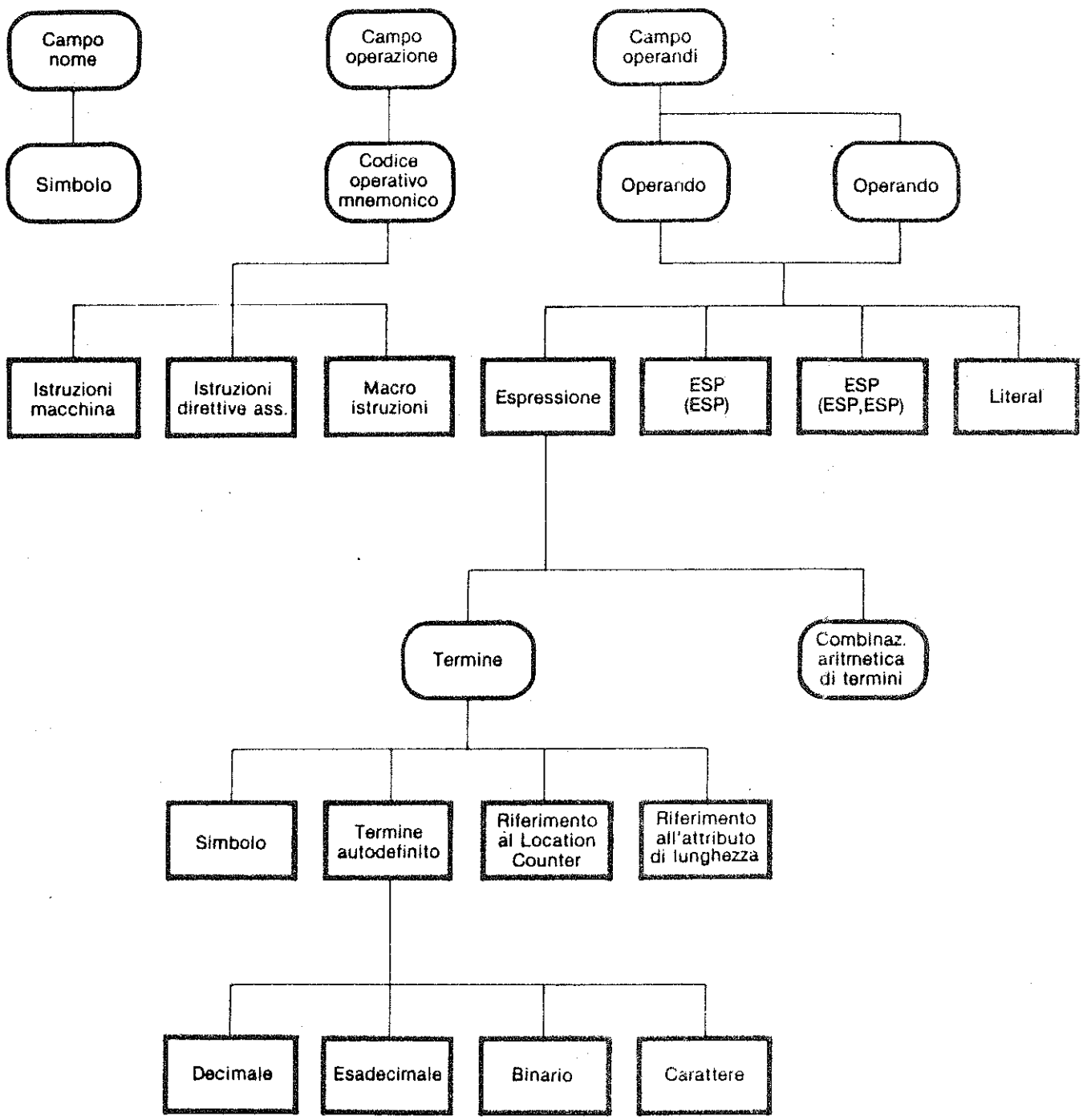
- TITLE
- EJECT
- SPACE
- PRINT.

Macro-istruzioni

Provocano l'inserimento di sequenze predefinite di istruzioni.

- PROC
- PRRET.

Struttura del linguaggio Assembler



Gli elementi di una frase Assembler

Il campo operandi di una frase Assembler è composto da una o più espressioni o literal; a sua volta una espressione è costituita da un termine o da una combinazione di termini.

I termini possono essere:

- rilocabili, quando il loro valore durante la fase di esecuzione dipende dal punto di caricamento del programma; devono quindi essere basati su qualche registro

- assoluti, quando il loro valore è indipendente dal punto di caricamento del programma.

Si distinguono 4 tipi di termini:

- simboli
- termini autodefiniti
- riferimenti al Location Counter
- riferimenti all'attributo lunghezza dei simboli.

Simboli

I simboli sono definiti e usati dall'utente per riferirsi a qualsiasi elemento del programma (aree di memoria, istruzioni, costanti, ecc.). Un simbolo è costituito al massimo di 8 caratteri alfanumerici il primo dei quali deve essere alfabetico.

L'Assembler associa ad ogni simbolo delle caratteristiche dette "attributi". Un simbolo viene detto "definito" quando compare nel campo nome di una frase sorgente; nell'ambito del modulo sorgente un simbolo è definibile una sola volta. Gli attributi che l'Assembler assegna ad un simbolo definito sono:

- valore
- lunghezza implicita
- tipo di simbolo.

Il valore assegnato ad un simbolo è l'indirizzo del primo byte di sinistra della zona di memoria contenente l'elemento in tal modo nominato. Il valore di un simbolo non può essere negativo e comunque superare $2^{24}-1$. Questo perché l'indirizzo di macchina viene espresso su 24 bit. La lunghezza implicita di un simbolo è la lunghezza, in byte, del campo di memoria nominato.

Un simbolo può essere rilocabile o assoluto in relazione al tipo di espressione presente nel campo operandi

della frase, che può per l'appunto essere rilocabile o assoluto. Il linguaggio Assembler richiede che nel campo operandi di particolari tipi di frasi compaiono simboli precedentemente definiti.

Esempio:

```
        ...
        ...
        ...
A       EQU     B
        ...
```

Questo significa che il simbolo B deve essere stato definito in una frase sorgente posta fisicamente prima dell'istruzione che lo riferisce.

Termini autodefiniti

Un termine autodefinito è quello il cui valore è costituito dal termine stesso; l'Assembler non assegna quindi alcun valore a questo termine. Tutti i termini autodefiniti sono considerati assoluti, poiché i loro valori non subiscono variazioni quando il programma di cui fanno parte viene caricato a differenti indirizzi di memoria. Il valore di un termine autodefinito non può superare i $2^{24}-1$ e il suo attributo lunghezza per definizione è 3 byte; i termini autodefiniti servono per specificare dati immediati, maschere, registri, incrementi di indirizzi, lunghezze esplicite, ecc.

Esempio:

```
MVI  A,X'Ø7'
```

l'attributo lunghezza del secondo operando vale 3 byte.

I termini autodefiniti si dividono in 4 tipi:

- decimale
- esadecimale
- binario.

<u>Termine autodefinito decimale</u>. E' un numero decimale senza segno, scritto come una sequenza di al più 8 caratteri numerici. L'Assemblatore provvede a convertire il dato da decimale in binario.

<u>Termine autodefinito esadecimale</u>. E' costituito da al più 6 caratteri esadecimali scritti tra apici (') e preceduti dalla lettera X.

Esempio:

```
X'A1BC'
```

Ogni carattere esadecimale è convertito in binario su 4 bit.

Termine autodefinito binario. E' una sequenza di caratteri zero e uno racchiusi tra apici e preceduti dalla lettera B.

Esempio:

B'11Ø1Ø1'.

Questo tipo di termine viene normalmente usato per definire maschere per operazioni logiche. Ogni carattere rappresenta un bit di memoria, e al massimo possono essere espressi 24 bit. Nel caso in cui il numero di caratteri non sia multiplo di 8, la normalizzazione con Ø binari è fatta sulla sinistra.

Termine autodefinito carattere. Questo tipo di termine è costituito da al più 3 caratteri (appartenenti alla tabella ISO) chiusi tra apici e preceduti dal carattere C.

Esempio:

C''
C'ABC'

Nota: I termini autodefiniti numerici (decimali, esadecimali e binari) vengono allineati a destra e riempiti con bit a zero a sinistra. I termini autodefiniti carattere vengono allineati a sinistra e riempiti con carattere spazio a destra.

Riferimento al Location Counter

Il programma Assembler gestisce un contatore di memoria o Location Counter che contiene, in ogni momento dell'assemblaggio, l'indirizzo del primo byte di memoria disponibile. Quando viene elaborata un'istruzione, sia che essa corrisponda ad una istruzione macchina, sia che essa definisca un'area dati, il Location Counter (eventualmente aggiustato per esigenze di allineamento) viene incrementato di un numero di byte pari alla lunghezza dell'elemento sviluppato (lunghezza dell'istruzione macchina o lunghezza dell'area dati). Se l'istruzione ha un nome, a tale simbolo viene assegnato il valore del Location Counter dopo l'eventuale aggiustamento, ma prima dell'incremento corrispondente alla lunghezza dell'istruzione stessa.

L'Assembler mantiene un Location Counter per ognuna delle sezioni di cui è composto il modulo sorgente cioè per la Sezione di Controllo (Control Section)

e per ognuna delle Sezioni Fantasma (Dummy Section). Il valore massimo del Location Counter è $2^{24}-1$. L'utente può riferirsi al valore corrente del Location Counter nel campo operandi delle frasi usando il carattere asterisco [*]: ciò equivale a dare un nome all'istruzione ed usare tale nome nel campo operandi. Quindi il termine [*] è un termine rilocabile.

L'attributo lunghezza del termine asterisco [*] può assumere i seguenti valori:

- lunghezza dell'istruzione macchina, quando è associato ad una istruzione esecutiva
- lunghezza della costante, quando è associato ad una istruzione di definizione dati
- 1, negli altri casi.

Il riferimento al Location Counter non deve essere utilizzato in una frase in cui si richieda l'uso di nomi predefiniti (salvo il caso di istruzioni ORG e EQU). L'eventuale uso del termine * nella definizione di dati nelle espressioni che indicano il fattore di duplicazione o il modificatore di lunghezza non dà luogo a segnalazione di errore: al termine viene attribuita la lunghezza Ø, e valore corrispondente al valore corrente del Location Counter senza l'eventuale aggiustamento per l'allineamento.

Riferimento all'attributo lunghezza dei simboli

L'attributo lunghezza di un simbolo può essere usato come termine mediante il prefisso L' (L maiuscolo seguito dal carattere apice) seguito dal simbolo stesso.

Esempio:

```
     ....
     ....
AFA  DS  F
BOY  DS  F
CAP  DS  CL8
     ....
MVC  BOY(L'AFA),CAP
```

L'attributo lunghezza implicito del simbolo AFA valorizza il suddetto termine (valore 4); si tratta quindi di un termine assoluto, il cui attributo lunghezza è 1. Cioè l'attributo lunghezza dell'espressione:

L'L'AFA

è pari a 1.

Espressioni

Un'espressione può essere costituita da un termine o da una combinazione di termini. Si distinguono due tipi di espressioni:

- semplici
- composte

Le espressioni composte devono seguire le normali regole del calcolo algebrico:

- ogni espressione può avere al massimo 16 termini

- gli operatori aritmetici ammessi sono: + - * /

- una espressione non può iniziare con un operatore aritmetico

- una espressione non può contenere 2 termini o 2 operatori aritmetici in successione

- combinazioni di termini racchiuse tra parentesi possono essere usate in combinazione con termini posti fuori dalle parentesi

- sono ammessi al massimo 5 livelli di parentesi.

Valutazione delle espressioni. Il calcolo delle espressioni viene eseguito da sinistra verso destra con la stessa priorità di esecuzione delle normali espressioni algebriche. La divisione per Ø è ammessa e dà come risultato Ø.

Attributi delle espressioni. L'attributo valore viene assegnato alla espressione durante il calcolo. L'attributo lunghezza è uguale all'attributo lunghezza del primo termine di sinistra della espressione.

Esempio:

```
   ALM A,B*C
   ...
B  DS  CL3
```

L'attributo lunghezza dell'espressione B*C è 3.

Le espressioni possono essere assolute o rilocabili:

- nel primo caso il valore rimane immutato a prescin-

dere dal punto di caricamento in memoria del programma

- nel secondo caso il valore dell'espressione subirà un mutamento (pari a N) se il programma che la contiene è rilocato in memoria di N byte rispetto alla originaria allocazione di memoria.

Un'espressione assoluta è costituita da un termine assoluto o da una espressione algebrica di termini assoluti.

Esempio:

```
A   DS   40
    ...
    ...
    ...
    MVI  A, X'A12'*12-B'00101001'
```

Può anche consistere di una combinazione aritmetica di termini rilocabili, eventualmente in combinazione con termini assoluti. In questo caso devono essere seguite le seguenti regole:

1. I termini rilocabili presenti nella espressione devono essere appaiati, cioè avere segno opposto e lo stesso attributo di rilocabilità (fare riferimento cioè allo stesso registro base). I termini appaiati possono anche non essere contigui. Nel seguito, al fine di sintetizzare gli esempi, si riportano solo i componenti dell'espressione

   RT → termine rilocabile
   AT → termine assoluto

   Esempio:

   RT + AT + RT -RT - AT - RT

2. I termini rilocabili non possono entrare in divisioni e moltiplicazioni:

   Esempio:

RT - RT * AT	non valida
(RT - RT) * AT	valida

   Quando appaiono termini rilocabili di segno oppo-

sto e con lo stesso attributo di rilocabilità, si annulla l'effetto della rilocazione e il valore dei termini rimasti appaiati rimane costante.

Un'espressione rilocabile è costituita da un termine rilocabile, oppure da una combinazione di termini rilocabili frammisti a termini assoluti; quest'ultima combinazione deve soddisfare le seguenti condizioni:

1. I termini rilocabili devono essere in numero dispari.

2. I termini rilocabili devono essere appaiati, tranne uno.

3. Il termine spaiato non deve essere preceduto dal segno meno.

A questo punto risulta chiaro che il valore dell'espressione corrisponde al valore del termine spaiato, e l'attributo di rilocabilità del termine spaiato diventa l'attributo di rilocabilità dell'espressione.

Esempio:

RT - (RT + RT - RT) + RT

Literal

E' possibile indicare, nel campo operandi di un'istruzione, dei literal, ovvero la rappresentazione di un dato anzichè il riferimento al dato stesso. I literal sono il mezzo più semplice per introdurre dei dati in un programma. Un literal è indicato con una costante preceduta dal segno uguale (=). La presenza di un literal all'interno di una frase sorgente fa sì che l'Assemblatore ponga il valore del literal stesso (cioè la costante specificata) in una zona di memoria denominata tavola dei literal, e sostituisca al literal, nella frase oggetto, l'indirizzo relativo. La definizione di un dato come literal segue le regole descritte per definizione di costanti.

Tavola dei literal. La tavola dei literal viene allocata alla fine della sezione di controllo a partire dall'indirizzo di Location Counter successivo a quello dell'ultima istruzione oggetto.

Il programma Assembler

Un programma eseguibile è costituito da una sequenza comunque complessa di istruzioni macchina. Perché un programma Assembler diventi eseguibile deve essere assemblato.

Per l'Assemblatore un programma è costituito da una Sezione di Controllo, ed eventualmente da una o più Sezioni Fantasma.

Sezione di Controllo (CONTROL SECTION). Una sezione di controllo è formata da tutte le frasi comprese fra una frase CSECT o START ed una frase END.

Ogni istruzione esecutiva incontrata viene tradotta nel corrispondente codice oggetto, ed ogni direttiva di definizione di aree o costanti provoca l'inclusione di tali campi nel codice oggetto.

Esempio:

```
      CSECT   2,Ø
      BALR    2,Ø
      USING   *,2
PAP   LA      3,4
      B       POP
PIP   DC      CL2'12'
POP   ST      3,8(2)
```

Questa sequenza di istruzioni provocherà, nell'Assemblaggio, la generazione dei seguenti 18 byte di codice oggetto:

0520
41300004
47F0200A
3132
50302008

Provocherà inoltre l'associazione dei seguenti attributi ai 3 simboli PAP, PIP, POP.

SIMBOLI	LUNGHEZZA	VALORE	TIPO
PAP	4	2	RILOCABILE
PIP	2	A	RILOCABILE
POP	4	C	RILOCABILE

Ogni riferimento ai campi PAP, PIP, POP verrebbe risolto dall'Assemblatore usando il registro base 2, così come indicato nell'istruzione USING.

Pertanto, se nel programma dovesse comparire una istruzione del tipo:

```
MVC  PIP,=C'22'
```

l'indirizzo di PIP verrebbe automaticamente generato dall'Assemblatore come '2008', cioè registro base 2 e scostamento pari ad '8'.

All'interno di una Sezione di Controllo possono essere comprese una o più Sezioni Fantasma.

Sezione Fantasma (DUMMY SECTION). Una Sezione Fantasma è iniziata da una frase DSECT terminata da una frase DSECT, CSECT o END. I simboli definiti in una Sezione Fantasma vengono riconosciuti dall'Assemblatore che associa a ciascuno di essi gli attributi (valore, lunghezza e tipo). A differenza di quanto avviene in una Sezione di Controllo, non viene però generato alcun codice oggetto.

L'utilità di una Sezione Fantasma è dunque proprio quella di definire formalmente una struttura senza che ciò dia luogo alla generazione di una sezione reale di codice.

Esempio:

```
      CSECT
      BALR    2,Ø
      USING   *,2
      USING   REC1,3
        :
TOC   LA      4,10
      LA      3,REC
MOVE  MVC     C1,K
      ALRI    3,12
        :
```

```
          :
          BCT     4,MOVE
          |

K         DC      CL3'AAA'
REC       DS      CL120
REC1      DSECT
CØ        DS      ØCL12
          DS      CL4
C1        DS      CL3
C2        DS      CL5
          END
```

Come già visto per la Sezione di Controllo, anche in questo caso l'Assemblatore genererebbe la seguente tabella di attributi per i simboli C0, C1, C2.

SIMBOLI	ATTRIBUTI		
	LUNGHEZZA	VALORE	TIPO
CØ	12	Ø	RILOCABILE
C1	3	4	RILOCABILE
C2	5	7	RILOCABILE

Il registro usato dall'Assemblatore per risolvere tutti i riferimenti a questi nomi è il registro generale 3 (USING REC1,3).

L'effetto di questo programma è di muovere la costante 'AAA' nei byte 5, 6 e 7 di ognuno dei 10 sottocampi di 12 byte del campo REC.

Infatti l'istruzione TOC carica nel registro 4 il valore del numero di volte per cui ripetere l'operazione (10). Si carica successivamente il registro 3 con l'indirizzo del campo REC. Si esegue quindi uno spostamento del contenuto del campo K ('AAA') nel campo C1. Il campo C1 non esiste come area perché faceva parte di una sezione fantasma, ne esiste però l'indirizzo che è dato dal contenuto del registro 3 più lo scostamento 4. Ma il contenuto del registro 3 è l'indirizzo del campo REC, per cui lo spostamento di 'AAA' avverrebbe nei byte 5, 6 e 7 del campo REC. Il contenuto del registro 3 è successivamente incrementato di 12, e si salta quindi all'istruzione MOVE dopo aver decrementato il contatore. L'istruzione MOVE non è

stata alterata; provocherà lo spostamento di 'AAA' ancora all'indirizzo '3004'. Questa volta il valore del registro 3 è pari all'indirizzo del campo REC aumentato di 12. La costante verrà pertanto spostata ai byte 17, 18 o 19 del campo REC.

Il processo si ripeterà provocando successivamente lo spostamento di 'AAA' nei byte 29, 30 e 31, e poi 41, 42 e 43, e così via per 10 volte.

Tutto ciò con solo 5 istruzioni, sfruttando le caratteristiche dello schema di indirizzamento e la definizione della struttura di un campo "fantasma" - REC1 - che in effetti non fa parte del codice.

3. LE ISTRUZIONI ASSEMBLER

Descrizione delle istruzioni Assembler

Nel seguito sono descritte dettaglatamente in ordine alfabetico le istruzioni e le macro-istruzioni del linguaggio Assembler P6Ø66. La descrizione segue uno schema fisso:

1. Descrizione funzionale: breve descrizione della funzione dell'istruzione.

2. Formato: descrizione del formato (dei formati) dell'istruzione.

3. Descrizione: descrizione dei campi nome operazione e operandi che compaiono nel formato dell'istruzione.

4. Espansione: descrizione delle istruzioni macchina generate dalla fase di macroespansione di una macro; la macroespansione avviene direttamente in codice oggetto.

Note: Le seguenti notazioni sono impiegate nella descrizione delle istruzioni ASSEMBLER, ma non appartengono al linguaggio:

{ } scelta obbligata di uno dei parametri indicati

[ ] inclusione opzionale di quanto specificato.

5. Esempi: esempi di impiego dell'istruzione descritta.

Istruzioni direttive per l'Assemblatore

Le istruzioni direttive per l'Assemblatore governano il processo di assemblaggio, e richiedono all'Assemblatore l'espletamento di funzioni ausiliarie. I loro formati e le loro descrizioni sono riportati nelle pagine seguenti.

CSECT

Descrizione Funzionale

Identifica l'inizio di una sezione di controllo. Tutte le frasi che seguono sono assemblate come facenti parte di questa sezione finchè non venga incontrata una frase che segni la fine della sezione o l'inizio di una nuova sezione o di un altro modulo sorgente (CSECT, DSECT, END).

Formato

nome	operazione	operando
[simbolo]	CSECT	

Descrizione

Nome

Se la frase è contrassegnata da un nome simbolico, questo indica l'indirizzo del 1° byte del modulo oggetto. A questo simbolo viene assegnato l'attributo lunghezza 1; esso è rilocabile.

Operandi

Il campo operandi non è usato e il Location Counter è inizializzato a Ø. Eventuali caratteri presenti nel campo operandi sono trattati come commento.

Esempio

```
SEZ      CSECT
         ...
         ...
         ...
         LA        1Ø, AREA
         USING     DEF, 1Ø
         ...
         ...
         ...
AREA     DS        CL40
         ...
         ...
         ...
         END
```

Descrizione Funzionale

Permette di introdurre in un programma dei dati sotto forma di costante.

Formato

nome	operazione	operando
[simbolo]	DC	[D] T [Ln] {'costante' / (costante)}

Descrizione

Nome

Se in questo campo è presente un simbolo, a quest'ultimo viene assegnato come valore l'indirizzo del 1° byte di sinistra della costante stessa e come attributo lunghezza la lunghezza implicita o esplicita della costante; si tratta di un simbolo rilocabile.

Operandi

Il campo operandi si compone di 4 sottocampi:

1. Sottocampo D - fattore di duplicazione; è un sottocampo opzionale; se presente provoca la riproduzione della costante per D volte, dopo che la costante è stata assemblata e completata nel suo esatto formato. Il fattore di duplicazione può essere espresso in due modi:

   - attraverso un termine autodefinito decimale

   - attraverso una espressione assoluta positiva chiusa tra parentesi, in cui gli eventuali simboli che compaiono siano precedentemente definiti.

Se il fattore di duplicazione è Ø, nessuna costante è

generata. L'istruzione DC con fattore di duplicazione Ø serve unicamente per assegnare al simbolo che compare nel campo nome un attributo lunghezza uguale alla lunghezza specificata nella frase senza riservare l'area relativa. Il fattore di duplicazione non è ammesso nella definizione delle costanti indirizzo.

2. Sottocampo T - tipo; è un sottocampo obbligatorio e indica all'assembler il tipo di costante da generare; questo sottocampo è costituito da una parola chiave.

Codice	Tipo di costante	Formato macchina
C	carattere	codice di 8 bit per ogni carattere specificato
X	esadecimale	codice di 4 bit per ogni carattere specificato
B	binaria	foramto binario; 1 byte ogni 8 caratteri specificati
F	binaria	formato binario: una parola
H	binaria	formato binario: una semiparola
S	indirizzo	valore del Registro base e dello scostamento; una semiparola

3. Sottocampo Ln - modificatore di lunghezza; è un sottocampo opzionale e se presente definisce la lunghezza in byte della costante. Può essere indicato nella forma:

   - termine autodefinito decimale

   - espressione assoluta positiva chiusa tra parentesi; i simboli che compaiono nella espressione devono essere precedentemente definiti.

Se la lunghezza esplicita non è definita, l'Assembler assume la lunghezza implicita della costante, che può dipendere dal valore costante specificato (X, B, C) o dal tipo della costante stessa (F, H). Nelle costanti che richiedono l'allineamento (F o H), specificare il modificatore comporta il non allineamento della costante stessa.

4. Sottocampo valore costante; questo sottocampo è obbligatorio e indica il valore della costante. Le costanti di dati (B/C/X F/H) devono essere scritte tra apici; le costanti indirizzo (S) devono essere racchiuse tra parentesi.

Costante carattere (Tipo C). Per definire il valore di una costante carattere può essere utilizzato uno qualsiasi dei caratteri della tabella ISO. Il valore costante deve essere racchiuso tra apici. Se il carattere apice (') deve far parte della costante, deve essere ripetuto due volte. Con la traduzione , nella stringa oggetto corrispondente apparirà un solo apice. La lunghezza massima della costante è di 256 byte. Se non è specificato il modificatore di lunghezza, l'Assembler assume come lunghezza della costante quella dedotta dal numero di caratteri indicati tra apici. Se è specificato il modificatore di lunghezza si avrà che:

- se il numero di caratteri della costante è inferiore alla lunghezza specificata, la costante è troncata a destra

- se il numero di caratteri della costante è inferiore alla lunghezza spacificata, la costante viene normalizzata a destra con spazi

Costante esadecimale (Tipo X). Le costanti esadecimali sono costituite da una sequenza di caratteri esadecimali (da Ø a 9, da A a F), racchiusa tra apici. La lunghezza massima è di 256 byte: quindi possono essere indicati fino a 512 caratteri esadecimali, dato che ogni coppia di caratteri esadecimali rappresenta un byte di memoria e viene quindi tradotta in un byte. Le costanti esadecimali sono allineate a destra nel campo. Pertanto, se il numero di cifre è dispari, il primo carattere esadecimale di sinistra darà origine ad un byte i cui 4 bit meno significativi centengono il digit esadecimale mentre i 4 bit più significativi sono a zero. Se non viene specificata la lunghezza, l'Assembler assume come lunghezza della costante la parte intera

dell'espressione $\frac{n+1}{2}$, dove n è il numero di caratteri esadecimali specificati. Se la lunghezza è specificata si ottiene che:

- se il numero delle coppie di caratteri esadecimali eccede la lunghezza specificata, i caratteri più a sinistra sono troncati

- se il numero delle coppie di cifre esadecimali è inferiore alla lunghezza specificata, i byte necessari per raggiungere tale lunghezza sono inseriti a sinistra e posti a zero binario.

Costante binaria (Tipo B). Le costanti binarie sono rappresentate mediante una sequenza di Ø e 1, racchiusa tra apici. La lunghezza massima è di 256 byte. La lunghezza implicita è data dal numero di byte occupati dalla costante, inclusa l'eventuale necessaria normalizzazione con bit a zero, poichè ogni carattere Ø o 1 rappresenta un bit della memoria; tale lunghezza implicita è pari alla parte intera dell'espressione $\frac{n+7}{8}$ dove n è il numero di cifre binarie.

Gli allineamenti nel campo sono fatti a destra; per la normalizzazione ed il troncamento valgono gli stessi principi che per le costanti esadecimali (X).

Costanti binarie (Tipo F e H). Contengono un numero decimale con o senza segno, racchiuso tra apici. Esso viene convertito nel corrispondente valore binario e messo in una parola (tipo F) o semiparola (tipo H) con l'allineamento appropriato: la lunghezza implicita della costante tipo F è quindi 4, due per la costante tipo H. Se è specificata la lunghezza, la quale non può essere superiore a 4, non viene eseguito l'allineamento. Se il valore del numero eccede la lunghezza esplicita od implicita della costante, il segno si perde e sono troncati i bit più a sinistra. Un valore negativo è posto nella forma di complemento a 2 del corrispondente valore positivo. I valori massimi delle costanti binarie sono:

lunghezza costante	Max	Min
4	$2^{31} - 1$	-2^{31}
2	$2^{15} - 1$	-2^{15}
1	$2^{7} - 1$	-2^{7}

Costanti indirizzo (S). Una costante indirizzo contiene un indirizzo di memoria, espresso in sorgente tramite una o più espressioni racchiuse fra parentesi. Il fattore di duplicazione non è ammesso.

Indirizzo tipo S: serve per ottenere indirizzi espressi nella forma registro base e scostamento. L'indirizzo può essere specificato:

1. Tramite una espressione rilocabile o assoluta.

2. Tramite 2 espressioni assolute, la prima delle quali rappresenta il valore di scostamento, la seconda il registro base.

Questo tipo di costante è assemblata in una mezza parola: i primi 4 bit di sinistra rappresentano il registro base; i rimanenti 12 bit esprimono lo scostamento. Se viene specificata una lunghezza esplicita, che non può essere diversa da 2, non viene effettuato l'allineamento.

Tabella riassuntiva delle costanti

Tipo	Lunghezza implicita in BYTE	Allineamento	Duplicazione	Gamma di lunghezza esplicita	Delimitatore	Valore della costante	Troncamento Normalizzazione
C	-	Byte	ammessa	$1 \div 256^N$	'	caratteri ISO	destra
X	-	Byte	ammessa	$1 \div 256^N$	'	crt esadecim.	sinistra
B	-	Byte	ammessa	$1 \div 256$	'	crt binari 1/Ø	sinistra
F	4	Parola	ammessa	$1 \div 4$	'	numero decim.	sinistra
H	2	semi parola	ammessa	$1 \div 4$	'	numero decim.	sinistra
S	2	semi parola	non ammessa	2	()	- una espressione assoluta o rilocabile - due espressioni assolute: exp(exp)	-

Nota Nelle istruzioni DS in cui sia specificato un valore costante di tipo C o X, il modificatore di lunghezza o la lunghezza implicita può essere anche superiore a 256, fino a un massimo di 65535.

Esempio

```
FILE   DC  C'END'        genera END
COST1  DC  3C'AZ'        genera AZAZAZ
COST2  DC  2CL5'AZ'      genera AZØØØAZØØØ
COST3  DC  2CL4'ABCDEF'  genera ABCDABCD
COST4  DC  C'C"ERA'      genera C'ERA

COST1  DC  X'A4BC7'      genera 0A4BC7 (3 byte)
COST2  DC  2XL4'A1F43'   genera ØØØA1F43ØØØA1F43
                                 8 byte)
COST3  DC  3XL2'A1F43'   genera 1F431F431F43 (6 byte)

COST   DC  B'11Ø1'          genera ØØØØ11Ø1 (1 byte)
COST1  DC  BL1'11Ø111ØØ1'   genera 1Ø111ØØ1 (1 byte)

DC  H'3Ø'  genera ØØ1E (2 byte)
DC  F'-3'  genera FFFFFFFD (4 byte)

DC  S(BETA)
DC  S(4ØØ(13))
```

DROP

Descrizione Funzionale

Permette di escludere un certo numero di Registri generali dal gruppo di registri precedentemente specificati quali registri base mediante frasi USING.

Formato

nome	operazione	operando
~~b~~	**DROP**	R_1 [, R_2 ..... R_n]

Descrizione

Nome

Non ammesso

Operandi

Questo campo contiene una successione di espressioni assolute, separate dalla virgola che specificano i registri base da sopprimere. Ogni espressione può avere valore da Ø a 15; il numero max di tali espressioni è 16.

Esempio

```
        SEZ     START   1Ø
                ...
                ...
                ...
                USING   BASE, 1Ø
                ...
                ...
                ...
A       BASE    DS      ØH
                ...
                ...
                ...
B               DROP    1Ø
                ...
                ...
                ...
C               USING   BASE, 1Ø
                ...
                ...
                ...
                END
```

Tutti i simboli definiti fra il punto A ed il punto B verrebbero risolti utilizzando il registro 10.

Fra il punto B ed il punto C invece non sarebbe possibile definire alcun simbolo a meno che nel programma non fosse specificato qualche altro registro base tramite un'altra istruzione USING.

Descrizione Funzionale

Permette di riservare aree di memoria, attribuendo loro eventualmente dei nomi per poterle indirizzare in modo simbolico.

Formato

nome	operazione	operando
[simbolo]	DS	[D] T [Ln] [('costante') / ((costante))]

Descrizione

Nome

Se in questo campo è presente un simbolo, a quest'ultimo viene assegnato l'indirizzo del 1° byte di sinistra dell'area stessa e come attributo lunghezza la lunghezza implicita o esplicita dell'area; si tratta di un simbolo rilocabile.

Operandi

Il campo operandi si compone di 4 sottocampi:

1. Sottocampo D - fattore di duplicazione; è un sottocampo opzionale, se presente provoca la riproduzione dell'area di memoria per D volte. Il fattore di duplicazione può essere espresso in 2 modi:

   - attraverso un termine autodefinito decimale

   - attraverso una espressione assoluta positiva chiusa tra parentesi, in cui gli eventuali simboli che compaiono siano precedentemente definiti.

Se il fattore di duplicazione è Ø, l'istruzione può essere usata per assegnare un nome ad un'area di memoria, senza che tale area sia riservata effettivamente. Altre frasi DS possono riservare l'area stessa, assegnando nomi ai suoi sottocampi. Quindi il fattore di duplicazione Ø serve ad assegnare al simbolo che compare nel campo nome un attributo lunghezza uguale alla lunghezza specificata nella frase, senza riservare l'area relativa.

2. Sottocampo T - tipo; è un sottocampo obbligatorio e indica all'Assembler il tipo di campo da generare; questo sottocampo è costituito da una parola chiave.

3. Sottocampo Ln - modificatore di lunghezza; è un sottocampo opzionale e se presente definisce la lunghezza in byte dell'area. Può essere indicato nella forma:

   - termine autodefinito decimale

   - espressione assoluta positiva chiusa tra parentesi; i simboli che compaiono nella espressione devono essere precedentemente definiti.

   La lunghezza max permessa per le definizioni di aree in cui sia specificato il tipo C o X è 65535 byte.

4. Sottocampo costante; è un sottocampo opzionale. Se presente, esso viene riportato nel listing e può concorrere alla determinazione della lunghezza del campo. Tale valore non genera il codice oggetto corrispondente. Le costanti dati (B/C/X/F/H) devono essere scritte tra apici; le costanti indirizzo (S) devono essere racchiuse tra parentesi.

Nota. Per ulteriori informazioni sulle costanti nei loro vari tipi, vedere istruzione DC.

Nel complesso le istruzioni DS sono trattate dall'Assembler come quelle DC, con le seguenti differenze:

- a fronte di istruzioni DS viene riservata un'area di memoria la cui lunghezza è:

  . specificata implicitamente dal tipo

. specificata esplicitamente tramite il fattore di duplicazione, il modificatore di lunghezza o il valore della eventuale costante

. pari a uno nel caso di istruzione DS del tipo C/X/B in cui non compaiono nè il modificatore, nè il valore costante

Le aree così riservate vengono azzerate.

<u>Esempio</u>

```
DS  CL5       viene riservata un'area di 5 byte
DS  C'ASMIS'  viene riservata un'area di 5 byte
DS  5X'AB'    viene riservata un'area di 5 byte.
```

DSECT

Descrizione Funzionale

Identifica l'inizio di una Sezione Fantasma (Dummy Section). In un modulo sorgente sono ammesse max 255 Sezioni Fantasma (Dummy Section). Una Sezione Fantasma (Dummy Section) si intende conclusa dal comparire di un'altra frase DSECT o CSECT.

Formato

nome	operazione	operando
Simbolo	**DSECT**	

Descrizione

Nome

E' obbligatorio; è un simbolo rilocabile il cui valore equivale all'indirizzo del 1° byte di tale sezione. L'attributo lunghezza è 1.

Operandi

Questo campo non è usato, ed eventuali caratteri presenti sono trattati come commento.

Esempio

```
                ...
                ...
                ...
                USING     DUM, 6
                ...
                ...
                ...
DUM             DSECT     CL3Ø
NAME            DS        CL5
COD             DS        CL4
IMPORT          DS
INIT            DSECT
                ...
                ...
                ...
```

I simboli 'NAME', 'COD' e 'IMPORT' verrebbero risolti usando il registro generale 6 come registro base.

EJECT

Descrizione Funzionale

Provoca la stampa della linea successiva del listing su una nuova pagina. Se l'istruzione viene riconosciuta all'inizio di una pagina di stampa, non viene eseguita.

Formato

nome	operazione	operando
b	EJECT	

Descrizione

Nome

Non ammesso.

Operandi

Non sono richiesti operandi, e quindi qualsiasi carattere compaia in questo campo è considerato come commento.

Esempio

```
        START   256
INIT    BALR    11, Ø
        USING   *, 11
        BC      15, GO
        ...
        ...
        ...
        EJECT
        ...
        ...
        ...
        END
```

END

Descrizione Funzionale

Provoca la fine dell'assemblaggio, e quindi deve essere l'ultima del modulo sorgente.

Formato

nome	operazione	operando
b	END	

Descrizione

Nome

Non ammesso

Operandi

Non sono usati, quindi qualsiasi carattere compaia in questo campo è considerato commento.

Esempio

```
SEZ      START    1Ø
         ...
         ...
         ...
         USING    BASE, 1Ø
         ...
         ...
         ...
BASE     DS       F
         ...
         ...
         ...
         END
```

Descrizione Funzionale

E' utilizzata per definire un simbolo a cui assegnare il valore e tutti gli attributi dell'espressione specificata nel campo operandi.

Formato

nome	operazione	operando
Simbolo	**EQU**	**Espressione**

Descrizione

Nome

E' obbligatorio.

Operandi

In questo campo è ammessa qualsiasi espressione assoluta o rilocabile. Ogni simbolo che compare in tale espressione, deve essere precedentemente definito. Al simbolo definito nel campo nome viene attribuito il valore e gli attributi lunghezza e di rilocabilità dell'espressione nel campo operandi.

Esempio

```
            START
RR          LR        3,4
RX          A         3, GIVEN
            ...
            ...
            ...
GIVEN       DC        F'33'
AREA        DS        XL2ØØØ
ONE         DS        CL24Ø
TWO         DS        CL8Ø
            ...
            ...
            ...
A           EQU       X'FF'
B           EQU       L'TWO
            ...
            ...
            ...
E           EQU       AREA+1ØØ
            ...
            ...
            ...
H           EQU       RX
            ...
            ...
            ...
            END
```

Descrizione Funzionale

Permette di modificare il contenuto del Location Counter della sezione a cui la frase appartiene.

Formato

nome	operazione	operando
b	ORG	[Espressione rilocabile]

Descrizione

Nome

Non ammesso

Operandi

Se in questo campo è presente una espressione rilocabile, al Location Counter della sezione che si sta assemblando viene assegnato il valore di tale espressione. Se il campo operando è omesso, l'istruzione non ha alcun effetto sul Location Counter.

Il simbolo che compare nell'espressione deve essere precedentemente definito, e deve essere definito nella stessa sezione in cui compare la ORG. Una istruzione ORG non può essere usata per specificare una allocazione precedente all'inizio della sezione in cui compare.

Esempio

```
          ...
CARD      DS    Ø CL8Ø
COD       DS      CL1Ø
NAME      DS      CL4Ø
GIVEN     DS      CL3Ø
          ORG     * - 8Ø
ART       DS      CL3Ø
LARGE     DS      CL7
          DS      CL43
          ...
          ...
          ...
          END
```

Il simbolo 'ART' verrebbe risolto con lo stesso indirizzo di 'CARD' e 'COD'.

PRINT

Descrizione Funzionale

Serve per controllare la stampa del listing dell'Assembler; più precisamente serve per specificare le modalità di stampa da quel punto in poi.

Formato

nome	operazione	operando
b	PRINT	[{ON / OFF}][{DATA / NO DATA}]

Descrizione

Nome

Non ammesso.

Operandi

Gli operandi possono essere indicati entrambi separati da virgola.

ON → stampa del listing
OFF → non avviene la stampa

DATA → le costanti vengono stampate per tutta la loro lunghezza nel listing

NO DATA → sono stampati solo i primi 8 byte a sinistra delle costanti

gli operandi di default sono: ON e NO DATA.

Quando si specifica OFF, il successivo operando non ha alcun effetto. Poichè in più punti del sorgente possono apparire istruzioni PRINT, se in una frase un operando è omesso, si assume che l'opzione relativa rimanga invariata rispetto all'ultima precedente specificazione.

Esempio

```
                ...
                ...
                ...
NUM             DS        D
TRAT            DC        PL6'Ø'
CIFRES          DC        PL5'Ø'
IMPON           DC        PL4'Ø'
ATTPRE          DC        PL7'Ø'
                ...
                ...
                ...
                PRINT     ON, NO DATA
                ...
                ...
                ...
                END
```

SPACE

Descrizione Funzionale

Ha la funzione di inserire nel listing una o più interlinee di spaziatura.

Formato

nome	operazione	operando
⊬	**SPACE**	**[valore decimale]**

Descrizione

Nome

Non ammesso

Operandi

Questo campo è opzionale; se non specificato viene eseguita una sola interlinea; se specificato vengono eseguite un numero di interlinee pari al valore decimale specificato nel campo. Se tale valore supera il numero delle righe rimanenti della pagina che si sta stampando, questa istruzione si comporta come l'istruzione EJECT.

Esempio

```
SEZ     START     1Ø
        ...
        ...
        ...
        USING     BASE, 1Ø
        SPACE     5
        ...
        ...
        ...
BASE    DS        D
        ...
        ...
        ...
        END
```

START

Descrizione Funzionale

Identifica l'inizio di una sezione di controllo. Tutte le frasi che seguono sono assemblate come facenti parte di questa sezione, finchè non venga incontrata una frase che segni la fine della sezione o l'inizio di una nuova sezione o di un'altro modulo sorgente (CSECT, DSECT, END).

Formatc

nome	operazione	operando
[simbolo]	START	[termine autodefinito]

Descrizione

Nome

Se la frase è contrassegnata da un nome simbolico, questo indica l'indirizzo del 1° byte del modulo oggetto. A questo simbolo viene assegnato l'attributo lunghezza 1; esso è rilocabile.

Operandi

E' un campo opzionale; se presente specifica il valore iniziale da assegnare al Location Counter della sezione. Se tale valore non è divisibile per 4, l'Assembler lo arrotonda per eccesso in modo da garantire l'allineamento dei vari elementi oggetto della sezione. Se l'operando è omesso (vedi CSECT) il Location Counter è settato a zero.

L'istruzione START non identifica l'inizio dell'Assemblaggio nè la prima sezione di controllo del programma ma semplicemente una sezione di controllo. L'istruzione START si differenzia dalla CSECT perchè in essa è consentito indicare il valore iniziale da assegnare al Location Counter (stampato nel listing).

Esempio

```
PROG1    START    2Ø4Ø
PROG2    START
PROG3    START    X'7F8'
         START
         ...
         ...
         ...
         END
```

TITLE

Descrizione Funzionale

Viene usata per identificare un listing Assembler. Provoca la stampa dell'intestazione voluta sulle pagine di tabulato che seguono fino alla lettura di un'altra frase TITLE. A fronte di questa frase viene eseguito il salto pagina.

Formato

nome	operazione	operando
b	**TITLE**	**'character list'**

Descrizione

Nome

Non ammesso

Operandi

Una sequenza di caratteri racchiusa tra apici. Possono essere espressi fino a 1ØØ caratteri, e se nella sequenza l'utente vuole esprimere il carattere apici deve indicarlo 2 volte.

Esempio

```
         START
PGM1     TITLE     'PRIMA COMPILAZIONE'
         ...
         ...
         ...
INIT     BALR      11,Ø
         USING     *, 11
         BC        15, GO
         ...
         ...
         ...
         END
```

USING

Descrizione Funzionale

Indica all'Assemblatore il Registro generale disponibile da utilizzare come Registro Base nelle istruzioni esecutive ed il valore di Location Counter che si intende associato a tale Registro.

Formato

nome	operazione	operando
b	USING	V, r_1 [, r_2, ... r_n]

Descrizione

Nome

La lettera V sta ad indicare un'espresssione assoluta o rilocabile; r1, r2, ... rn sono espressioni assolute che designano i registri generali, quindi il valore di queste espressioni deve essere compreso tra Ø ÷ 15, e ne sono consentite al max 16.

In seguito a questa istruzione l'Assemblatore assume che il registro generale r1 sia caricato con il valore dell'espressione assoluta o rilocabile V, r_2 sia caricato con il valore del registro r_1 + 4096 .... ed r_n con il valore del registro generale r_1 + (n-1)*4096.

Un'istruzione USING che riferisca un registro già menzionato in una precedente frase USING ne annulla l'effetto.

Se l'operando 1 nella frase USING è il registro generale Ø, l'Assemblatore assumerà che tale registro contenga zero qualunque sia il valore di V.

Un programma che usi il registro Ø come registro base non è rilocabile. Se in un programma è specificato più

di un registro base ed un certo simbolo cade nel campo di indirizzabilità di più di uno di essi, per la sua traduzione viene utilizzato il registro base che produce il minore displacement.

Esempio

```
LARGE    START    Ø
         BALR     9, Ø
         USING    POD, 9, 1Ø, 12
POD      LM       1Ø, 11, BASE
         B        INIT
BASE     DC       A (POD+4Ø96)
         DC       A (POD+8192)
INIT     LA       3, 1 (14)
         ...
         ...
         ...
         END
```

Macro-istruzioni

Le macro-istruzioni vengono interpretate dall'Assemblatore, che sostituisce ad ognuna di esse un gruppo predefinito di istruzioni assembler, detto espansione della macroistruzione.

I formati e le descrizioni della macro-istruzioni sono riportati nelle pagine seguenti.

Descrizione Funzionale

Questa macro-istruzione serve per generare il prologo del modulo, cioè per allocare al modulo in questione un'area LOCALE per il salvataggio dei registri d'uso per le aree di lavoro. Se presente deve immediatamente seguire l'istruzione CSECT/START.

Formato

nome	operazione	operando
[simbolo]	PROC	[RX], [RY] [, ENDLOC = simbolo 1] [, STARLOC = simbolo 2] [, PROLOG = {YES / NO}]

Descrizione

Nome

E' opzionale; se presente corrisponde al nome del modulo oggetto. Se assente al modulo viene assegnato il nome della CSECT/START.

Operandi

- RX e RY sono operandi posizionali opzionali. Se presenti vengono espressi mediante termini autodefiniti decimali il cui valore deve essere compreso nell'intervallo Ø ÷ 15. Indicano gli estremi del gruppo di registri generali il cui contenuto deve essere salvato nell'area LOCALE (i registri sono considerati contigui agli estremi cioè il registro Ø è considerato successivo al registro 15).

- ENDLOC = simbolo 1; è un operando opzionale a parola chiave che specifica, se presente, il nome del 1° byte successivo all'area locale del modulo. Se non indicato l'Assembler assume che tale nome sia ENDLOCAL.

- STARLOC = simbolo 2; è un operando opzionale a parola chiave che specifica il nome dell'area locale. Se non indicato, viene assunto come nome dell'area locale LOCAL.

- PROLOG = $\left\{\frac{\text{YES}}{\text{NO}}\right\}$; è un operando opzionale che specifica all'Assembler se il prologo deve essere generato o no. Il valore di default è PROLOG = YES.

Espansione

```
[Simbolo]  LR    Ø, 13
           LA    13, {ENDLOCAL / Simbolo 1} - {LOCAL / Simbolo 2}
           ASA   13, 13
           ST    Ø, Ø (, 13)
          [STM   RX, RY, 4 (13)]
```

Le istruzioni macchina elencate nella espansione sono valorizzate dagli operandi espressi nella frase sorgente. Nel caso della frase STM, essa viene generata solo se nella frase sorgente sono indicati gli operandi RX e RY.

Esempi

1. POP PROC ,,STARLOC = BEGLOC

   Viene generata la seguente sequenza di istruzioni:

```
POP   LR    Ø, 13
      LA    13, ENDLOCAL - BEGLOC
      ASA   13, 13
      ST    Ø, Ø (, 13)
```

2. POP PROC 2, 3

   Viene generata la seguente sequenza:

```
POP   LR    Ø, 13
      LA    13, ENDLOCAL - LOCAL
      ASA   13, 13
      ST    Ø, Ø (, 13)
      STM   2, 3, 4 (13)
```

Descrizione Funzionale

Questa macro-istruzione serve per generare l'epilogo del modulo; essa viene espansa con le istruzioni complementari rispetto a quelle generate dalla PROC. L'area LOCALE viene deallocata dallo stack e viene ripristinato il contenuto dei registri di lavoro eventualmente specificati nella PROC.

Formato

nome	operazione	operando
[simbolo]	**PRRET**	**[I]**

Descrizione

Nome

Può contenere un simbolo.

Operandi

Può essere indicato opzionalmente un termine autodefinito decimale il cui valore deve essere compreso nell'intervallo Ø ÷ 255. Questo termine indica il displacement di rientro. Se omesso si assume che il rientro abbia displacement Ø.

Espansione

```
[Simbolo] [LM      RX, RY, 4 (13)]
          L        13, Ø (, 13)
          LA       Ø, { ENDLOCAL Simbolo 1 } - { LOCAL Simbolo 2 }
          FSA      Ø, Ø
          RETEXT   [I]
```

Se precedentemente non è stata generato un prologo (PROC), l'espansione della PRRET è:

```
[nome]    RETEXT    [I]
```

Esempio

Se nel prologo generato dalla PROC non era incluso il salvataggio dei registri, viene omessa l'istruzione di ripristino dei registri (LM).

```
PUP     L       13, Ø (, 13)
        LA      Ø, ENDLOCAL - BEGLOC
        FSA     Ø, Ø
        RETEXT
```

Descrizione Funzionale

Somma del contenuto di una parola al contenuto di un registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	A	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	5A

Azione

Il contenuto dell'area di memoria indirizzata dal 2° operando (una parola) è addizionato al primo operando ed il risultato è posto nel primo operando. L'operazione è eseguita sommando i 32 bit dei due operandi, segni compresi. Un overflow si verifica quando i riporti del segno e del bit di ordine più alto sono discordanti (riporto del segno uguale a 1 e riporto della posizione più alta uguale a Ø, o viceversa).

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " < Ø
2 " " " " " " > Ø
3 se si verifica un'overflow

Esempio

A 2,100 (,4)

dove:

1° operando reg. 2 il cui valore esadecimale è Ø1Ø8BA

2° operando 1ØØ(,4) " " " " " ØØ68

risultato reg. 2 " " " " " Ø1Ø922

1° operando = R1

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ 1Ø111Ø1Ø

2° operando = parola

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ Ø11Ø1ØØØ

risultato = R1

1ØØØØØØØ	ØØØØØØØ1	1ØØØ1ØØ1 ØØ1ØØØ1Ø

Descrizione Funzionale

Somma il contenuto di una mezza parola al contenuto di un registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	AH	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	4A

Azione

Il contenuto della semiparola indirizzata dal secondo operando è addizionato al contenuto del primo operando e la somma è posta nel primo operando. Prima dell'operazione il segno della semiparola viene espanso per una lunghezza di 16 bit; l'operazione è quindi eseguita sommando i 32 bit dei due operandi segni compresi. Un overflow si verifica quando i riporti del segno e del bit di ordine più alto sono discordanti (riporto del segno uguale a 1 e riporto della posizione più alta uguale a Ø, o viceversa).

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " Ø
2 " " " " " " Ø
3 se si verifica un'overflow

Esempio

AH 2,50 (2,5)

dove:

1° operando reg. 2 il cui valore esadecimale è Ø1Ø8BA

2° operando 5Ø(2,5) " " " " " ØØ68

risultato reg. 2 " " " " " Ø1Ø922

1° operando = R1

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ 1Ø111Ø1Ø

2° operando = mezza parola con espansione del segno

ØØØØØØØØ	ØØØØØØØØ	ØØØØØØØØ Ø11Ø1ØØØ

ESPANSIONE

risultato = R1

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØ1 ØØ1ØØØ1Ø

Descrizione Funzionale

Somma logica del contenuto di una parola al contenuto di un registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	AL	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	5E

Azione

Il contenuto del campo indirizzato dal secondo operando è addizionato al contenuto del primo operando ed il risultato è posto nel primo operando. L'operazione è eseguita sommando i 32 bit dei due operandi. Se si verifica un riporto il Condition Code viene posto ai seguenti valori:

C.C.: Ø se il risultato è = Ø e non c'è riporto
1 " " " " ≠ Ø " " " "
2 " " " " = Ø e c'è riporto
3 " " " " ≠ Ø " " "

Esempio

AL 3,100 (,4)

dove:

1° operando reg. 3 il cui valore esadecimale è Ø1Ø8BA

2° operando 1ØØ(,4) " " " " " ØØ8Ø68

risultato reg. 3 " " " " " Ø18922

1° operando = R1

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ 1Ø111Ø1Ø

2° operando = parola di memoria

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ Ø11Ø1ØØØ

risultato = R1

1ØØØØØØØ	ØØØØØØØ1	1ØØØ1ØØ1 ØØ1ØØØ1Ø

Descrizione Funzionale — Somma logica di due campi di memoria.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	ALM	D_1 (L_1, B_1), D_2 (L_2, B_2) S_1 (L_1), S_2 (L_2) S_1, S_2 D_1 (, B_1), D_2 (, B_2)	SS	FB

Azione

Il campo di memoria indirizzato dal secondo operando è sommato al campo di memoria indirizzato dal primo operando. Il risultato viene posto nel primo operando. Se il secondo operando è più corto del primo, esso viene "logicamente" esteso con zeri a sinistra. Gli operandi vengono allineati a destra e l'operazione procede da destra a sinistra per il numero di byte indicato in L1.

C.C.: Ø se il risultato è = Ø senza riporto
1 " " " " ≠ Ø " "
2 " " " " = Ø con riporto
3 " " " " ≠ Ø " "

Esempio

ALM 50 (2,4), 20 (1,13)

dove:

1° operando 5Ø(2,4) il cui valore esadecimale è Ø8BA

2° operando 2Ø(1,13) " " " " " ØØ68

risultato 5Ø(2,4) " " " " " Ø922

1° operando = campo di memoria

ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando = campo di memoria

ØØØØØØØØ	Ø11Ø1ØØØ

ESTENSIONE LOGICA

risultato = campo di memoria

ØØØØ1ØØ1	ØØ1ØØØ1Ø

Descrizione Funzionale

Somma logicamente il contenuto di un registro al contenuto di un altro registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	ALR	R_1, R_2	RR	1E

Azione

Il contenuto del registro R2 è sommato al contenuto del registro R1 ed il risultato è posto in R1. L'operazione è eseguita sommando i 32 bit dei due operandi. Se si verifica un riporto dalla posizione del segno, il Condition Code viene posto ai seguenti valori.

C.C.: Ø se il risultato è = Ø e non c'è riporto
1 " " " " ≠ Ø " " " "
2 " " " " = Ø e c'è riporto
3 " " " " ≠ Ø " " "

Esempio

ALR 4,5

dove:

1° operando reg. 4 il cui valore esadecimale è Ø1Ø8BA

2° operando reg. 5 " " " " " ØØ8Ø68

risultato reg. 4 " " " " " Ø18922

1° operando = R1

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ 1Ø111Ø1Ø

2° operando = R2

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ Ø11Ø1ØØØ

risultato = R1

1ØØØØØØØ	ØØØØØØØ1	1ØØØ1ØØ1 ØØ1ØØØ1Ø

ADD LOGICAL REGISTER IMMEDIATE

Descrizione Funzionale — Somma un immediato ad un registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	ALRI	R_1, I_2	RR	01

Azione — Somma al contenuto del registro R1 (1° operando) l'immediato indicato nei 4 bit costituenti il 2° operando. Questa somma non dà segnalazioni di overflow ma si segnala l'eventuale riporto.

C.C.: 1 se la somma senza riporto è $\neq \emptyset$
2 " " " con " " $= \emptyset$
3 " " " " " " $\neq \emptyset$

Esempio

ALRI 5,X'03

dove:

1° operando reg. 5 il cui valore esadecimale è Ø1Ø8BA

2° operando x'Ø3' " " " " " Ø3

risultato reg. 5 " " " " " Ø1Ø8BD

1° operando = R1

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando = I2

ØØ11

risultato = R1

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø1111Ø1

Descrizione Funzionale — Somma algebrica di due campi di memoria.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	AM	D_1 (L_1, B_1), D_2 (L_2, B_2) S_1 (L_1), S_2 (L_2) S_1, S_2 D_1 (, B_1), D_2 (, B_2)	SS	FA

Azione

Il contenuto del campo di memoria indirizzato dal secondo operando è addizionato al contenuto del campo di memoria indirizzato dal primo operando. L'addizione è fatta in modo algebrico, cioè tenendo conto del segno e il risultato è posto nel primo operando. Il bit del segno è il bit più pesante del byte di sinistra del campo su cui si opera. Il campo più corto viene logicamente esteso espandendo il bit del segno e l'operazione procede dal byte meno pesante a quello più pesante. Si verifica overflow:

- se i riporti del segno e del bit di ordine più alto sono discordanti

- se il campo in cui viene depositato il risultato non è sufficiente a contenerlo.

In caso di overflow gli L1 bytes meno pesanti del risultato vengono assegnati al campo di arrivo.

C.C.: Ø se il risultato è = Ø
1 " " " " < Ø
2 " " " " > Ø
3 se si verifica overflow

Esempio **AM 100 (2,4), 50 (1,6)**

dove:

1° operando 1ØØ(2,4) il cui valore esadecimale è Ø8BA

2° operando 5Ø(1,6) " " " " " ØØ68

risultato 1ØØ(2,4) " " " " " 8922

1° operando = campo di memoria

ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando = campo di memoria

ØØØØØØØØ	Ø11Ø1ØØØ

ESPANSIONE LOGICA DEL SEGNO

risultato = campo di memoria

1ØØØ1ØØ1	ØØ1ØØØ1Ø

Descrizione Funzionale

Somma il contenuto di un registro al contenuto di un altro registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	AR	R_1,R_2	RR	1A

Azione

Il contenuto del registro R2 è sommato al contenuto del registro R1 ed il risultato è posto in R1. L'operazione è eseguita sommando i 32 bit dei due operandi, segni compresi. Un overflow si verifica quando i riporti del segno e del bit di ordine più alto sono discordanti (riporto del segno uguale ad 1 e riporto della posizione più alta uguale a Ø, o viceversa). Ponendo R1 = R2 si ottiene come risultato il raddoppio del contenuto del registro.

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " < Ø
2 " " " " " " > Ø
3 se si verifica un overflow

Esempio

AR 5,4

dove:

1° operando reg. 5 il cui valore esadecimale è Ø1Ø8BA

2° operando reg. 4 " " " " " ØØ8Ø68

risultato reg. 5 " " " " " Ø18922

0° operando = R1

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ 1Ø111Ø1Ø

2° operando = R2

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ Ø11Ø1ØØØ

risultato = R1

1ØØØØØØØ	ØØØØØØØ1	1ØØØ1ØØ1 ØØ1ØØØ1Ø

Descrizione Funzionale — Alloca un'area sullo stack.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	ASA	R_1, R_2	RR	23

Azione

Alloca sullo stack un'area pari al numero di byte indicato dal contenuto del registro R2 e passa nel registro R1 l'indirizzo aggiornato della cima dello stack (TOP OF STACK). Se il contenuto del registro R2 è Ø non viene allocata alcuna area ed in R1 si ottiene il valore della cima della stack.

C.C.: non gestito

Descrizione Funzionale — Salto e memorizzazione di indirizzo.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	BAL	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	45

Azione — Memorizza nel registro R1 l'indirizzo della prossima istruzione successiva alla BAL stessa ed esegue un salto all'indirizzo espresso dal secondo operando.

C.C.: non gestito

Descrizione Funzionale — Salto e memorizzazione di indirizzo.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	BALR	R_1, R_2	RR	05

Azione

Ha le stesse funzioni della BAL. L'indirizzo a cui viene compiuto il salto è espresso nel registro R2.

Se R2 indica il registro generale Ø l'esecuzione procede in sequenza con l'istruzione successiva alla BALR.

C.C.: non gestito

Descrizione Funzionale

Esame del valore del Codice Condizione e salto condizionato all'indirizzo definito nell'istruzione.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	BC	$M_1, D_2 (X_2, B_2)$ $M_1, D_2 (, B_2)$ $M_1, S_2 (X_2)$ M_1, S_2	RX	47

Azione

La maschera presente nella istruzione permette di esaminare il contenuto del Codice Condizione, e spedifica per quali valori si esegue il salto. Se il contenuto del Codice Condizione è diverso dai valori specificati nella maschera viene eseguita la prossima istruzione in sequenza.

Esiste la seguente corrispondenza fra valori del Codice Condizione e maschere

C.C.		Maschera
0	→	8
1	→	4
2	→	2
3	→	1

Esempio:

BC 8, COLM salta se C.C. = Ø

E' possibile eseguire l'esame di più valori del Codice Condizione specificando una maschera pari alla somma delle singole maschere.

Esempio:

BC 9, COLM salta per C.C. = Ø oppure C.C. = 3

Un valore di maschera pari a 15 provoca l'effettuazione del salto per qualunque valore del Codice Condizione.

Nel seguito si dà un elenco dei codici mnemonici che permettono di scrivere i salti condizionati senza dovere specificare i valori della maschera. I codici mnemonici estesi sono tradotti dall'Assemblatore nelle corrispondenti combinazioni di codice di operazione (BC o BCR) e maschera.

CODICE ESTESO	SIGNIFICATO	ISTRUZIONE ASSEMBLER EQUIVALENTE
B D2(X2,B2)	Branch	BC 15,D2(X2,B2)
BR R2	Branch(RR format)	BCR 15,R2
NOP D2(X2,B2)	No Operation	BC Ø,D2(X2,B2)
NOPR R2	No operation(RR format)	BCR Ø,R2
BH D2(X2,B2)	Branch on High	BC 2,D2(X2,B2)
BL D2(X2,B2)	Branch on Low	BC 4,D2(X2,B2)
BE D2(X2,B2)	Branch on Equal	BC 8,D2(X2,B2)
BNH D2(X2,B2)	Branch on Not High	BC 13,D2(X2,B2)
BNL D2(X2,B2)	Branch on Not Low	BC 1Ø,D2(X2,B2)
BNE D2(X2,B2)	Branch on Not Equal	BC 6,D2(X2,B2)
BO D2(X2,B2)	Branch on Overflow	BC 1,D2(X2,B2)
BP D2(X2,B2)	Branch on Plus	BC 2,D2(X2,B2)
BM D2(X2,B2)	Branch on Minus	BC 4,D2(X2,B2)
BZ D2(X2,B2)	Branch on Zero	BC 8,D2(X2,B2)
BNP D2(X2,B2)	Branch on Not Plus	BC 13,D2(X2,B2)
BNM D2(X2,B2)	Branch on Not Minus	BC 10,D2(X2,B2)
BNZ D2(X2,B2)	Branch on Not Zero	BC 6,D2(X2,B2)
BNO D2(X2,B2)	Branch on Not Ones	BC 14,D2(X2,B2)

Descrizione Funzionale

Esame del Codice Condizione e salto condizionato all'indirizzo definito nell'istruzione.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	BCR	M_1, R_2	RR	07

Azione

La maschera presente nella istruzione permette di esaminare il contenuto del Codice Condizione, e specifica per quali valori si esegue il salto. Se il contenuto del Codice Condizione è diverso dai valori specificati nella maschera viene eseguita la prossima istruzione in sequenza (per quanto riguarda l'uso ed i valori della maschera, vedere istruzione BC).

Descrizione Funzionale

Salto per registro diverso da zero previo decremento di uno del suo contenuto.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	BCT	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	46

Azione

Il contenuto del registro R1 viene diminuito di 1; se il valore risultante è diverso da zero viene eseguito un salto all'istruzione il cui indirizzo è espresso dal secondo operando. L'indirizzo di salto viene determinato prima della diminuzione del registro R1. Il salto avviene anche se il registro R1 contiene un numero negativo. Nel caso che il registro contenga il massimo numero negativo rappresentabile in un registro, dopo la diminuzione conterrà il massimo numero positivo rappresentabile in un registro essendosi inoltre verificato il salto. L'overflow relativo non è segnalato. Se in partenza R1 contiene Ø, il salto si verifica dopo di che il registro vale -1.

C.C.: non gestito

Descrizione Funzionale

Salto per registro diverso da zero, previo decremento di uno del suo contenuto.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	BCTR	R_1, R_2	RR	06

Azione

Il contenuto del registro R1 viene diminuito di 1; se il valore risultante è diverso da zero viene eseguito un salto all'istruzione il cui indirizzo è espresso dal secondo operando. L'indirizzo di salto viene determinato prima della diminuzione del registro R1. Se si pone R2 = Ø il salto non avviene, ed il contenuto del registro R1 è diminuito di 1. Il salto avviene anche se il registro R1 contiene un numero negativo. Nel caso che il registro contenga il massimo numero negativo, dopo la diminuzione conterrà il massimo numero positivo essendosi inoltre verificato il salto. L'overflow relativo non è segnalato. Se in partenza R1 contiene Ø, il salto si verifica ed il registro vale -1.

C.C.: non gestito

Descrizione Funzionale

Salto se la somma del contenuto di due registri è maggiore del contenuto di un terzo.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	BXH	$R_1, R_2, D_3 (B_3)$ R_1, R_2, S_3	RS	86

Azione

Il registro R1 (indice) viene incrementato col valore contenuto nel registro R2 (incremento). Il contenuto di R1 viene poi confrontato col contenuto di un terzo registro immediatamente seguente R2 (limite). Se il contenuto dell'indice è maggiore del limite, si salta all'istruzione il cui indirizzo è espresso dal terzo operando; se è minore o uguale si procede in sequenza. L'indirizzo di salto viene determinato prima dell'incremento del registro indice. Tutti i bit dei registri interessati partecipano all'operazione. Le quantità che intervengono sono considerate numeri binari (positivi o negativi).

C.C.: non gestito

Descrizione Funzionale

Salto se la somma del contenuto di due registri è minore o uguale al contenuto di un terzo.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	BXLE	$R_1, R_2, D_3 (B_3)$ R_1, R_2, S_3	RS	87

Azione

Il registro R1 (indice) viene incrementato col valore contenuto nel registro R2 (incremento). Il contenuto di R1 viene poi confrontato col contenuto di un terzo registro immediatamente seguente R2 (limite). Se il contenuto dell'indice è $\leqslant$ del limite, si salta alla istruzione il cui indirizzo è espresso dal terzo operando; se è $>$ si procede in sequenza. L'indirizzo di salto viene determinato prima dell'incremento del registro indice. Tutti i bit dei registri interessati partecipano all'operazione. Le quantità che intervengono sono considerate numeri binari (positivi o negativi).

C.C.: non gestito

Descrizione Funzionale

Confronta algebricamente il contenuto di un registro con il contenuto di una parola.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	**C**	**R_1, D_2 (X_2, B_2)** **R_1, D_2 (, B_2)** **R_1, S_2 (X_2)** **R_1, S_2**	**RX**	**59**

Azione

Il contenuto del registro R1 è confrontato algebricamente col contenuto della parola indirizzata dal secondo operando. Il risultato del confronto determina lo stato del Condition Code.

C.C.: Ø se contenuto di R1 = contenuto secondo operando
1 " " " " < " " "
2 " " " " > " " "

Descrizione Funzionale

Salto con memorizzazione dell'indirizzo di rientro e passaggio della lista degli argomenti.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	CALL	$D_1 (B_1), D_2 (L_2, B_2)$ $S_1, S_2 (L_2)$ S_1, S_2 $D_1 (B_1), D_2 (,B_2)$ $D_1 (B_1) [(\ldots D_n (B_n))]$ $S_1 [(\ldots S_n)]$	SI	9A

Azione

Il primo operando è l'indirizzo della istruzione a cui eseguire il salto. Il secondo operando (1 o più di 1) costituisce la lista degli argomenti. Dopo la esecuzione della istruzione, il registro generale 1 punta ad un'area di stack lunga n+2 parole (n è il numero degli argomenti) contenenti rispettivamente e nell'ordine:

1. prima parola: nel byte più pesante il numero di argomenti (n) e nei tre meno pesanti l'indirizzo della istruzione di rientro seguente la CALL.

2. seconda parola: informazioni per il sistema operativo.

3. Se n è diverso da zero seguono n parole contenenti ciascuna rispettivamente, il primo byte uguale a zero e gli altri tre byte l'indirizzo del primo, del secondo ... dell'ennesimo argomento di chiamata della CALL.

Se la CALL viene eseguita a seguito di un EXEC l'indirizzo memorizzato per primo nella lista di n+2 parole è quello dell'istruzione seguente l'EXEC.

Se n = Ø il registro generale 1 punta ad una lista di 2 parole contenente solo gli elementi dei punti 1 e 2.

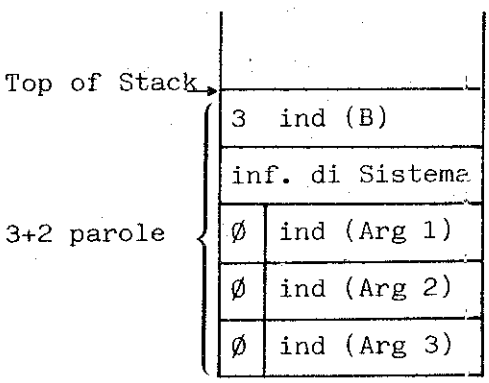
C.C.: non gestito.

Esempio

L'istruzione:

```
        ...
        CALL    X,Arg 1,Arg 2,Arg 3
B       EQU     *
        ...
```

Provocherebbe, in esecuzione, il passaggio del controllo all'indirizzo corrispondente al simbolo 'X' dopo aver modificato lo Stack nel seguente modo:



Descrizione Funzionale

Converte il contenuto di una mezza parola nel corrispondente numero decimale in caratteri ISO.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	CBS	D_1 (L, B_1), D_2 (B_2) S_1 (L), S_2 S_1, S_2 D_1 (, B_1), D_2 (, B_2)	SS	DB

Azione

Il primo operando indirizza il campo di memoria in cui andranno despositati i caratteri generati dalla conversione. Il secondo operando indirizza la mezza parola il cui contenuto è da convertire. Alla fine della conversione il registro generale RØ contiene il numero di caratteri generati. Se si esaurisce il campo di lunghezza L prima di aver terminato la conversione, il campo di arrivo contiene i primi L caratteri generati e il registro generale RØ il valore L.

C.C.: Ø conversione effettuata
1 " "
2 " "
3 esaurito il campo di lunghezza L prima della fine della conversione.

Esempio

2° operando mezza parola = 41

rappresentazione binaria ØØØØØØØØ Ø1ØØØØØ1
rappresentazione decimale 65
rappresentazione decimale in caratteri ISO A

risultato in 1° operando = 36 35

Descrizione Funzionale

Converte un numero dalla rappresentazione DATA alla rappresentazione ISO.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	**CDS**	**D_1 (, B_1), D_2 (B_2)** **S_1, S_2** **S_1, S_2** **D, (, B_1) D_2 (, B_2)**	**SS**	**DA**

Azione

Converte il dato numerico di tipo DATA indirizzato dal secondo operando, in formato sorgente (o ISO) nel campo indirizzato dal primo operando.

Il registro generale Ø contiene la lunghezza massima del campo d'arrivo. Il numero viene allineato a sinistra nel campo. Dopo la conversione con esito positivo il registro generale Ø contiene il numero di caratteri residui nel campo di partenza, il registro generale 1 contiene l'indirizzo del primo byte non interessato dalla conversione (nel campo di partenza) e il registro generale 2 contiene il numero di caratteri generati. Se il campo d'arrivo non può contenere il numero, il campo d'arrivo e i registri Ø, 1 e 2 restano inalterati.

Un'eventuale specificazione di lunghezza viene accettata dall'Assemblatore ma non provoca alcun effetto.

C.C.: Ø = conversione corretta
1 = non possibile
2 = errore nel formato DATA
3 = campo d'arrivo troppo piccolo

Descrizione Funzionale

Converte un numero dalla rappresentazione DATA alla rappresentazione Memoria.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	**CDM**	**D_1 (B_1), D_2 (B_2)** **S_1, S_2** **S_1, S_2** **D_1 (, B_1), D_2 (, B_2)**	**SS**	**DD**

Azione

Converte il dato numerico in formato DATA indirizzato dal secondo operando, in formato Memoria nel campo indirizzato dal primo operando. La lunghezza del campo di partenza è definita implicitamente nel campo stesso.

Il tipo di formato di Memoria (semplice o doppia precisione) è indicato nel primo semibyte del campo ricevente. Una cifra da Ø a 9 determina una conversione del tipo semplice precisione; D o B determinano la doppia precisione.

Alla fine della conversione il registro generale 1 contiene l'indirizzo del byte successivo all'ultimo interessato dalla conversione nel campo di partenza.

C.C.: non gestito.

Descrizione Funzionale

Confronta algebricamente il contenuto di un registro con una mezza parola.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	CH	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	49

Azione

Il contenuto del Registro R1 è confrontato algebricamente con la mezza parola (estesa per propagazione del segno a sinistra) contenuta nel secondo operando.

C.C.: Ø se contenuto di R1 = contenuto secondo operando
1 " " " " < " " "
2 " " " " > " " "

Descrizione Funzionale

Confronta logicamente il contenuto di un registro con il contenuto di una parola.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	CL	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	55

Azione

Il contenuto di R1 è confrontato logicamente al contenuto della parola indirizzata dal secondo operando. Il risultato del confronto determina lo stato del Codice Condizione. I termini vengono confrontati bit a bit da sinistra verso destra (segno compreso), l'operazione si arresta quando vengono trovati due bit diversi. In tal caso viene considerato maggiore quello dei due termini al quale appartiene il bit 1.

C.C.: Ø se contenuto R1 = contenuto secondo operando
1 " " " < " " "
2 " " " > " " "

Descrizione Funzionale — Confronta logicamente due campi di memoria.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	CLC	D_1 (L, B_1), D_2 (B_2) S_1 (L), S_2 S_1, S_2 D_1 (, B_1), D_2 (, B_2)	SS	D5

Azione

Il contenuto del campo indirizzato dal primo operando è confrontato logicamente con il contenuto del campo indirizzato dal secondo operando. Il confronto inizia da sinistra e procede, byte per byte, verso destra, per la lunghezza specificata al primo operando. Il confronto non tiene conto del segno degli operandi. L'operazione si arresta non appena vengono trovati due bit diversi. Il risultato del confronto determina lo stato del Codice Condizione.

C.C.: Ø se 1° operando = 2° operando
1 " " " < " "
2 " " " > " "

Descrizione Funzionale

Confronta un byte di memoria con un byte contenuto nell'istruzione.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	CLI	D_1 (B_1), I_2 S_1, I_2	SI	95

Azione

Il contenuto del byte indirizzato dal primo operando è confrontato logicamente con l'immediato dell'istruzione. Il risultato del confronto determina lo stato del Codice Condizione. Il confronto avviene come per l'istruzione CLC.

C.C.: Ø se contenuto del 1° operando = I2
1 " " " " " < I2
2 " " " " " > I2

Descrizione Funzionale — Confronta due campi di memoria.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	CLM	D_1 (L_1, B_1), D_2 (L_2, B_2) S_1 (L_1), S_2 (L_2) S_1, S_2 D_1 (, B_1), D_2 (, B_2)	SS	F5

Azione

La zona di memoria indirizzata dal 1° operando viene confrontata con con la zona di memoria indirizzata dal 2° operando. Il risultato dell'istruzione determina lo stato del Codice Condizione. Gli operandi vengono allineati a destra, e l'operazione procede da sinistra a destra per il numero di byte indicato in L1. Se il secondo operando è più corto viene esteso con zeri a sinistra.

C.C.: Ø i 2 operandi sono uguali
1 il 1° operando è < del 2°
2 il 1° operando è > del 2°

COMPARE LOGICAL REGISTER

CLR

Descrizione Funzionale — Confronta logicamente il contenuto di due registri.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	CLR	R_1, R_2	RR	15

Azione

Il contenuto del registro R1 è confrontato logicamente con il contenuto del registro R2. Il risultato della operazione determina lo stato del Codice Condizione. I due termini vengono confrontati bit per bit da sinistra verso destra (segno compreso) e l'operazione si arresta in presenza di due bit diversi (Ø ed 1). Viene considerato maggiore quello dei due termini al quale appartiene il bit a 1.

C.C.: Ø se contenuto R1 = contenuto R2
1 " " " < " "
2 " " " > " " .

Descrizione Funzionale — Confronto algebrico di due campi di memoria.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	CM	D_1 (L_1, B_1), D_2 (L_2, B_2) S_1 (L_1), S_2 (L_2) S_1, S_2 D_1 (, B_1), D_2 (, B_2)	SS	F9

Azione

La zona di memoria indirizzata dal 1° operando è confrontata algebricamente con la zona di memoria indirizzata dal 2° operando; il risultato dell'operazione determina lo stato del Condition Code. L'operando più corto viene espanso logicamente con il bit del segno a sinistra. Gli operandi vengono allineati a destra e il confronto procede da sinistra a destra. Nella comparazione il bit più pesante del byte più pesante di un campo è considerato bit di segno del campo.

C.C.: Ø se i due operandi sono uguali
1 se 1° operando < 2° operando
2 " " " > " "

Descrizione Funzionale

Converte un numero dalla sua rappresentazione in formato Memoria a quella in formato sorgente (o ISO) secondo formato.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	CMSS	D_1 (, B_1), D_2 (B_2) S_1, S_2 S_1 (L), S_2 S_1, S_2 D_1 (, B_1), D_2 (, B_2)	SS	DF

Azione

Converte il dato numerico in formato Memoria indirizzato dal secondo operando, in formato ISO nel campo indirizzato dal primo operando secondo l'indicazione di formato contenuta nel byte meno pesante del registro generale 2.

- Il registro generale Ø stabilisce le dimensioni massime del campo ricevente.

- il registro generale 1 contiene, dopo la conversione, l'indirizzo del primo byte del campo di arrivo non interessato dalla conversione (il numero viene allineato a sinistra nel campo d'arrivo).

- Dopo la conversione il registro generale Ø contiene il numero di byte residui nel campo d'arrivo. Se il campo d'arrivo non è sufficiente a contenere il risultato della conversione i contenuti del campo di arrivo, dei registri generali Ø, 1 e 2 sono invariati.

- Il valore del byte meno pesante del registro generale 2 stabilisce il formato di conversione secondo le seguenti regole:

Contenuto Registro Generale 2	Caratteristiche del numero dopo la conversione
Ø	arrotondato ad intero
1	arrotondato ad 1 decimale
2	arrotondato a 2 decimali
3	arrotondato a 3 decimali
4	arrotondato a 4 decimali
5	arrotondato a 5 decimali
6	arrotondato a 6 decimali
7	arrotondato a 7 decimali
8	arrotondato a 8 decimali
9	arrotondato a 9 decimali
10	arrotondato a 10 decimali
11	arrotondato a 11 decimali
12	arrotondato a 12 decimali
13	con 13 cifre decimali
14	secondo il formato dell'istruzione PRINT del linguaggio BASIC
15	Floating point (una cifra, il punto decimale, fino a 18 cifre l'esponente).

Un'eventuale specificazione della lunghezza L viene accettata dall'Assemblatore ma non ha alcun effetto.

C.C.: Ø = conversione compiuta
1 = conversione compiuta
2 = conversione compiuta
3 = campo d'arrivo più piccolo del necessario.

Descrizione Funzionale

Converte un numero dalla rappresentazione in formato Memoria a quella in formato Sorgente (o ISO) secondo formato

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	CMST	D_1 (B_1), D_2 (B_2) S_1, S_2 S_1 (L), S_2 S_1, S_2 D_1 (, B_1), D_2 (, B_2)	SS	DE

Azione

Converte il dato numerico in formato Memoria indirizzato dal secondo operando, in formato ISO nel campo indirizzato dal primo operando secondo le regole del formato standard di output (vedi istruzione PRINT del BASIC). Il registro genrale Ø stabilisce le dimensioni massime del campo d'arrivo.

Il numero viene allineato a sinistra nel campo d'arrivo. Dopo la conversione, se questa ha avuto esito positivo, il registro generale Ø contiene il numero di byte residui del campo d'arrivo, il registro 1 contiene l'indirizzo del carattere successivo al campo convertito ed il registro 2 contiene, nel byte meno significativo, il numero di caratteri generati. Se il campo d'arrivo non è sufficiente a contenere il risultato della conversione, il campo d'arrivo e i registri generali Ø, 1 e 2 restano inalterati.

C.C.: Ø = conversione corretta
1 = conversione corretta
2 = conversione corretta
3 = campo d'arrivo troppo piccolo.

Descrizione Funzionale — Confronta il contenuto di due registri.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	CR	R_1, R_2	RR	19

Azione

Il contenuto di R1 è confrontato algebricamente con il contenuto di R2. Il risultato del confronto determina lo stato del Condition Code.

C.C.: Ø se il contenuto di R1 = al contenuto di R2
1 " " " " " < " " " "
2 " " " " " > " " " "

Descrizione Funzionale

Converte una stringa di caratteri ISO numerici nel corrispondente numero binario.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	CSBH	D_1 (B_1), D_2 (L_2, B_2) S_1, S_2 (L_2) S_1, S_2 D_1 (B_1), D_2 (, B_2)	SS	F6

Azione

Converte in binario senza segno la stringa di caratteri numerici indirizzati dal secondo operando nella semiparola indirizzata dal primo operando. Il numero di byte da convertire è quello specificato dal verificarsi della prima delle seguenti eventualità:

- dopo N cifre si incontra un carattere diverso da cifra; $N < L2$

- dopo L2 cifre.

Il risultato della conversione è modulo 2^{16} (65536) ed è posto nel primo operando. R1 alla fine dell'operazione contiene l'indirizzo del primo byte della stringa che non è stato convertito.

Se il primo carattere della stringa non è numerico, il primo operando non viene modificato e R1 riceve l'indirizzo del secondo operando. Il numero da convertire deve essere $\leqslant$ 99999. Se è maggiore il primo operando non viene modificato e R1 riceve l'indirizzo del secondo operando.

C.C.: Ø se numero $\leqslant$ 65535
1 " " $>$ "

2 se numero > 99999
3 secondo operando non numerico

Esempio

2° operando = 36 35

rappresentazione decimale in caratteri ISO A
rappresentazione decimale → 65
rappresentazione binaria ØØØØØØØØ Ø1ØØØØØ1

risultato in 1° operando = 41

Descrizione Funzionale

Converte una stringa di caratteri ISO nel formato interno di tipo DATA.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	CSD	D_1 (B_1), D_2 (B_2) S_1, S_2 S_1, S_2 D_1 (, B_1), D_2 (, B_2)	SS	DB

Azione

Il primo operando indirizza il risultato della conversione. Il secondo operando indirizza il campo da convertire. Il registro generale Ø contiene, nel byte meno significativo, il numero massimo di caratteri da analizzare.

Se la conversione ha esito positivo il registro generale 1 contiene l'indirizzo del byte che ha fatto arrestare la conversione, il registro generale 2 contiene il numero di byte generati nella conversione mentre il registro generale Ø contiene il numero di byte sorgenti residui.

Se la conversione non viene effettuata, i registri generali Ø, 1 e 2 restano inalterati.

La conversione viene arrestata al primo carattere diverso da +,-,E,., o le cifre da Ø a 9 in accordo con la sintassi.

C.C.: Ø = conversione effettuata
1 = conversione effettuata
2 = conversione effettuata
3 = conversione non effettuata.

Descrizione Funzionale

Converte un numero dalla sua rappresentazione in formato Memoria a quella in formato Sorgente (o ISO) secondo formato.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	**CUMS**	**D_1 (, B_1) D_2 (B_2)** **S_1, S_2** **S_1 (L), S_2** **S_1, S_2** **D_1 (, B_1), D_2 (, B_2)**	**SS**	**D8**

Azione

Converte il dato numerico in formato Memoria indirizzato dal secondo operando, in formato ISO nel campo indirizzato dal primo operando secondo l'indicazione di formato contenuta in un campo indirizzata dal registro generale 1.

Il registro generale Ø contiene la lunghezza del campo ricevente. Se la specifica di formato indirizzata dal registro generale 1 è corretta (vedasi regole di formazione dell'istruzione IMAGE di BASIC), viene eseguita la conversione e si restituisce nello stesso registro 1 l'indirizzo del primo carattere del campo formato che non è stato utilizzato, nel registro Ø il numero di caratteri residui e nel registro 2 il numero di caratteri generati.

Se durante la conversione si esaurisce il campo d'arrivo o la specifica di formato è incorretta, l'esecuzione viene terminata.

Del registro generale 2 viene considerato solo il byte meno significativo.

Se la conversione si ferma con C.C. = 1 il campo di arrivo, i registri generali Ø, 1 e 2 restano inalte-

rati. Se la conversione termina con C.C. = 2 allora il registro generale 1 contiene l'indirizzo del byte successivo all'ultimo del campo formato errato.

Un'eventuale specificazione della lunghezza L viene accettata dall'Assemblatore ma non ha alcun effetto.

C.C.: Ø = conversione corretta
1 = Formato non numerico
2 = Formato errato
3 = Campo d'arrivo troppo piccolo.

Descrizione funzionale

Ricerca tabellare eseguita in modo dicotomico su un sottocampo di un elemento della tabella.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	DIS	D_1 (L_1, B_1), D_2 (L_2, B_2) S_1 (L_1), S_2 (L_2) S_1, S_2 D_1 (, B_1), D_2 (, B_2)	SS	F8

Azione

Viene fatto un confronto fra un sottocampo della tabella indirizzata dal primo operando e il campo di memoria indirizzato dal secondo operando. La ricerca è dicotomica e la relazione d'ordine è stabilita sul sottocampo da ricercare dalla numerazione binaria pura (senza segno). L1 è la lunghezza di un elemento della tabella. L2 è la lunghezza del secondo operando e quindi anche del sottocampo da confrontare. RØ contiene il numero totale degli elementi della tabella (solo gli ultimi 2 byte a destra sono significativi). R2 contiene l'offset del sottocampo nell'elemento. Si considerano solo i 4 bit meno pesanti di R2. Come risultato della ricerca viene caricato in R1 lo indirizzo dell'elemento ricercato se la ricerca ha esito positivo. Se la ricerca ha esito negativo viene caricato in R1 l'indirizzo dell'elemento immediatamente seguente quello ricercato. Se RØ = Ø si pone il Condition Code a 1 e in R1 l'indirizzo della tabella (1° operando).

C.C.: Ø se l'elemento è trovato =
1 " " " " ≠ e maggiore ultimo
2 " " " " ≠ e minore primo
3 " " " " ≠

Descrizione Funzionale

Divide il contenuto dei registri e memorizza risultato e resto.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	DM	$R_1, R_2, D_3 (B_3)$ R_1, R_2, S_3	RS	85

Azione

Il contenuto dei 2 byte meno significativi del registro R2, viene diviso per il contenuto dei 2 byte meno significativi del registro R1. Il risultato su 2 byte, seguito dal resto anch'esso su 2 byte, è posto in memoria all'indirizzo indicato dal terzo operando. Se i 2 byte meno significativi di R1 sono Ø, il risultato è 65535 ed il resto è il contenuto dei 2 byte meno significativi di R2. I contenuti di R1 e R2 non sono alterati dall'operazione.

C.C.: non gestito

Descrizione Funzionale

Consente l'esecuzione di una singola istruzione situata al di fuori della normale sequenza.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	EX	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	44

Azione

L'istruzione il cui indirizzo è indicato secondo operando viene alterata dal contenuto (ultimi 8 bit) del registro R1 ed eseguita. L'alterazione riguarda il solo byte della istruzione che segue il codice operativo (bit 8 ÷ 15), e avviene mediante la somma logica del contenuto del byte suddetto e del contenuto dell'ultimo byte di destra del registro R1 (bit 24 ÷ 31). Questa somma sostituisce il contenuto del byte dell'istruzione solo agli effetti dell'interpretazione e dell'esecuzione, infatti l'istruzione resta inalterata in memoria. Se R1 indica il registro generale Ø non avviene alcuna alterazione.

Dopo l'esecuzione della EXECUTE, e della istruzione alla quale la EXECUTE rimanda, il programma procede con l'istruzione immediatamente seguente la EXECUTE. Tutto questo, a meno che l'istruzione fuori sequenza non sia un salto, nel qual caso il contenuto del contatore della istruzione conterrà l'indirizzo di branch, ed il programma procederà dalla istruzione alla quale il salto ha passato il controllo. Se l'istruzione, il cui indirizzo è indicato nel secondo operando e una BAL, BALR o CALL l'indirizzo che viene salvato è quello della istruzione seguente la EXECUTE.

C.C.: gestito dalla istruzione fuori sequenza

Descrizione Funzionale — Dealloca un'area dello stack.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	FSA	R_1, R_2	RR	22

Azione

Dealloca dalla cima dello stack un'area pari al numero di byte indicato dal contenuto di R2 e si ritorna in R1 l'indirizzo della cima dello Stack (TOS). Di R2 si considerano solo i 2 byte meno significativi.

Se R1 indica il registro generale Ø, il suo contenuto non viene alterato.

C.C.: non gestito.

Descrizione Funzionale

Ritorna in un registro l'indirizzo di un argomento lasciato sullo Stack da un'istruzione del tipo CALL dal Linguaggio BASIC.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
simbolo	**GSA**	**R_1, R_2**	**RR**	**24**

Azione

Il byte meno significativo del registro R2 contiene il numero d'ordine dell'argomento di cui si vuol conoscere l'indirizzo (1 per l'ennesimo argomento ed n per il primo).

L'indirizzo dell'argomento viene registrato nel registro R1 per un massimo di 24 bit. I restanti 8 bit sono posti a zero.

Se il registro R2 contiene il valore ZERO, nei tre byte meno significativi di R1 viene caricato l'indirizzo di rientro dopo la CALL, mentre nel byte più pesante viene ritornato il numero di argomenti di chiamata.

C.C.: non gestito.

Descrizione Funzionale

Inserisce il contenuto di un byte di memoria in un registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	IC	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	43

Azione

Gli 8 bit del carattere indirizzato dal secondo operando sono inseriti nei bit meno significativi (24÷31) del registro R1. I rimanenti bit rimangono invariati.

C.C.: non gestito.

Descrizione Funzionale

Muove in memoria un immediato per il numero di volte indicato nel registro generale Ø.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	IM	D_1 (B_1), I_2 S_1, I_2	SI	9D

Azione

Muove in memoria all'indirizzo espresso dal primo operando, l'immediato (secondo operando) per il numero di byte indicato dai due byte meno pesanti del registro generale Ø. Se i due byte meno pesanti del registro Ø contengono Ø l'immediato viene scritto 65536 volte.

C.C.: non gestito

Descrizione Funzionale — Determina la categoria a cui appartiene un carattere.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	ISO	D_1 (X_1, B_1) D_1 (, B_1) S_1 (X_1) S_1	RX	82

Azione — Confronta il carattere indirizzato con la tabella ISO.

I caratteri alfabetici sono: A, B, C.....Z
I caratteri numerici sono : Ø, 1, 2.....9
Gli operatori sono : +, -, *, /, ↑

Dopo l'operazione il valore del Codice Condizione indica la categoria di appartenenza del carattere esaminato secondo i seguenti valori:

C.C.: Ø carattere = alfabetico
1 carattere = numerico
2 carattere = operatore
3 caratteri rimanenti

Descrizione Funzionale

Caricamento del contenuto di una parola in un registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	**L**	**R_1, D_2 (X_2, B_2)** **R_1, D_2 (, B_2)** **R_1, S_2 (X_2)** **R_1, S_2**	**RX**	**58**

Azione

Il contenuto della parola il cui indirizzo è specificato nel secondo operando, è caricato nel registro R1.

C.C.: non gestito.

Esempio **L 5,50 (4,3)**

dove:

1° operando reg. 5 il cui valore esadecimale è Ø1Ø8BA

2° operando 5Ø(4,3) " " " " " ØØ8Ø68

risultato reg. 5 " " " " " ØØ8Ø68

1° operando = R1

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ 1Ø111Ø1Ø

2° operando = parola

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ Ø11Ø1ØØØ

risultato in R1 =

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ Ø11Ø1ØØØ

Descrizione Funzionale — Carica in un registro generale un indirizzo.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	LA	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2 R_1, I_2	RX	41

Azione

L'indirizzo del secondo operando viene caricato per un massimo di 24 bit nel registro generale R1. Il registro viene caricato da destra; la parte non utilizzata è riempita con zeri.

C.C.: non gestito

Descrizione Funzionale

Carica un registro con il complemento a due del contenuto di una parola di memoria.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	LC	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	53

Azione

Il complemento a due del contenuto della parola indirizzata dal secondo operando è caricato nel registro R1. Se il secondo operando contiene zero, tale valore viene caricato in R1 (con segno +). L'overflow è possibile solo nel caso si esegua il complemento del massimo numero negativo (-2^{31}).

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " < Ø
2 " " " " " " > Ø
3 se si verifica un overflow

Esempio

LC 6,100 (,4)

dove:

1° operando reg. 6 il cui valore esadecimale è Ø1Ø8BA

2° operando 1ØØ(,4) " " " " " ØØ8Ø68

risultato reg. 6 " " " " " FF7F98

1° operando = R1

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando = parola

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ	Ø11Ø1ØØØ

risultato in R1 =

11111111	11111111	Ø1111111	1ØØ11ØØØ

Descrizione Funzionale

Carica un registro con il complemento a due di un altro registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	LCR	R_1, R_2	RR	13

Azione

Il complemento a due del contenuto del registro R2 è caricato nel registro R1. Se R2 contiene zero, tale valore viene caricato in R1 (col segno +). Se R1 = R2 il contenuto di R1 viene cambiato di segno. L'overflow è possibile solo nel caso si esegua il complemento a 2 del massimo numero negativo (-2^{31}) che può essere contenuto in un registro.

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " < Ø
2 " " " " " " > Ø
3 se si verifica un'overflow

Esempio

LCR 3,2

dove:

1° operando reg. 3 il cui valore esadecimale è Ø1Ø8BA

2° operando reg. 2 " " " " " ØØ8Ø68

risultato reg. 3 " " " " " FF7F98

1° operando = R1

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando = R2

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ	Ø11Ø1ØØØ

risultato in R1 =

11111111	11111111	Ø1111111	1ØØ11ØØØ

Descrizione Funzionale

Carica il contenuto di una mezza parola in un registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	LH	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	48

Azione

Il contenuto della mezza parola il cui indirizzo è dato dal secondo operando è caricato nel registro R1. Il segno della mezza parola (bit di maggior peso) viene espanso fino alla posizione del segno del registro.

C.C.: non gestito.

Esempio

LH 7,26 (2,10)

dove:

1° operando reg. 7 il cui valore esadecimale è Ø1Ø8BA

2° operando 26(2,1Ø)" " " " " ØØØØ68

risultato reg. 7 " " " " " ØØØØ68

1° operando R1 =

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando = mezza parola con espansione del segno

ØØØØØØØØ	ØØØØØØØØ	ØØØØØØØØ	Ø11Ø1ØØØ

ESPANSIONE

risultato in R1 =

ØØØØØØØØ	ØØØØØØØØ	ØØØØØØØØ	Ø11Ø1ØØØ

Descrizione Funzionale — Ricerca di un carattere in un campo.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	LIE	D_1 (B_1), I_2 S_1, I_2	SI	99

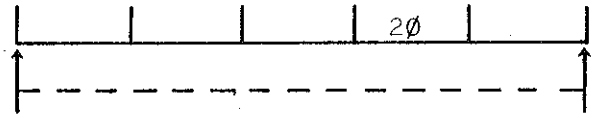
Azione

Viene ricercato il primo carattere uguale all'immediato sul campo di memoria il cui indirizzo è contenuto nel primo operando. La lunghezza massima per cui effettuare la ricerca è implicitamente espressa nel registro generale Ø, di cui si considerano solo i 2 byte meno pesanti. Se la ricerca ha esito positivo il Codice Condizione è posto a Ø, e viene passato nel registro R1 l'indirizzo del carattere uguale all'immediato. Nel registro Ø si trova la lunghezza residua del campo di memoria. Se la ricerca si esaurisce senza esito positivo, il Codice Condizione viene posto a 3, il registro generale Ø viene posto a Ø ed il registro generale 1 contiene l'indirizzo del primo byte di memoria successivo al campo analizzato. Se la lunghezza di ricerca contenuta nel registro Ø è uguale a Ø, il Codice Condizione viene posto a 3, e nel registro generale 1 viene depositato l'indirizzo del campo di memoria specificato come primo operando.

C.C.: Ø se la ricerca dà esito positivo
1 " " " " " "
2 " " " " " "
3 se la ricerca dà esito negativo

Immaginiamo di avere:

RØ = 5 e LIE Ø(6), x'2Ø'



Viene passata in RØ la lunghezza residua del campo; benchè il puntatore punti all'inizio del sottocampo in cui risiede il carattere da ricreare, la lunghezza passata è 1 e non 2.

Descrizione Funzionale

Ricerca del primo carattere diverso da un valore specificato.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	LINE	D_1 (B_1), I_2 S_1, I_2	SI	9B

Azione

Viene ricercato il primo carattere diverso dall'immediato nel campo di memoria il cui indirizzo è contenuto nel primo operando. La lunghezza massima per cui effettuare la ricerca è espressa dal valore del registro Ø, di cui si considerano solo i due byte meno pesanti. Se la ricerca ha esito positivo il Condition Code è posto a Ø, e in R1 viene depositato lo indirizzo del carattere trovato diverso dall'immediato. In RØ si trova la lunghezza residua del campo di memoria. Se la ricerca si esaurisce perchè si esaurisce il campo di memoria, il Codice Condizione viene posto a 3, il registro generale Ø viene posto a Ø ed il registro generale 1 contiene l'indirizzo del primo byte di memoria successivo al campo analizzato. Se la lunghezza di ricerca, contenuta nel registro Ø è uguale a Ø, viene posto:

- il Codice Condizione a 3
- nel registro R1 l'indirizzo del campo di memoria.

C.C.: Ø se la ricerca dà esito positivo
1 " " " " " "
2 " " " " " "
3 se la ricerca dà esito negativo

Descrizione Funzionale

Carica N registri adiacenti con il contenuto di N parole consecutive.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	**LM**	**R_1, R_2, D_3 (B_3)** **R_1, R_2, S_3**	**RS**	**98**

Azione

Il gruppo di registri consecutivi aventi per estremi i registri R1 e R2 è caricato col contenuto di altrettante parole, a partire dall'indirizzo contenuto nel secondo operando. I registri sono caricati in ordine ascendente, tenendo conto che il registro Ø è consecutivo al registro 15. Se R1 = R2 solo un registro viene caricato.

C.C.: non gestito

Descrizione Funzionale

Carica un registro con il contenuto, reso negativo, di una parola.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	LN	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	51

Azione

Il registro R1 viene caricato con il complemento a due del valore assoluto del contenuto della parola indirizzata dal secondo operando. Se il contenuto della parola è zero, questo valore resta immutato e dotato di segno positivo.

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " ≠ Ø

Esempio

LN 11,50 (,6)

dove:

1° operando reg. 11 il cui valore esadecimale è Ø1Ø8BA

2° operando 5Ø(,6) " " " " " ØØ8Ø68

risultato reg. 11 " " " " " FF7F98

1° operando = R1

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando = parola

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ	Ø11Ø1ØØØ

risultato in R1 =

11111111	11111111	Ø1111111	1ØØ11ØØØ

Descrizione Funzionale

Carica un registro con il contenuto, reso negativo, di un altro registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	LNR	R_1, R_2	RR	11

Azione

Il registro R1 è caricato con il complemento a 2 del valore assoluto del contenuto del registro R2. Se R2 contiene Ø, questo valore resta immutato e dotato di segno positivo (anche R1 conterrà poi Ø positivo). Se R1 = R2, il contenuto di R1 viene reso negativo.

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " < Ø

Esempio

LNR 9,11

dove:

1° operando reg. 9 il cui valore esadecimale è Ø1Ø8BA

2° operando reg. 11 " " " " " ØØ8Ø68

risultato reg. 9 " " " " " FF7F98

1° operando R1 =

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando R2 =

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ	Ø11Ø1ØØØ

risultato in R1 =

11111111	11111111	Ø1111111	1ØØ11ØØØ

LOAD POSITIVE REGISTER

LPR

Descrizione Funzionale

Carica un registro con il valore di un altro registro reso positivo.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	LPR	R_1, R_2	RR	10

Azione

Il valore assoluto del contenuto di R2 viene posto in R1. Se il contenuto di R2 è negativo, l'operazione è preceduta da una complementazione a 2. L'overflow è possibile solo quando viene eseguita da una complementazione del massimo numero negativo contenuto in un registro (-2^{31}). Se R1 = R2 il contenuto di R1 viene reso positivo. Nel caso di massimo numero negativo l'istruzione non viene eseguita.

C.C.: Ø se il risultato dell'operazione è = Ø
2 " " " " " " > Ø
3 se si verifica un'overflow

Esempio

LPR 13,6

dove:

1° operando reg. 13 il cui valore esadecimale è Ø1Ø8BA

2° operando reg. 6 " " " " " ØØ8Ø68

risultato reg. 13 " " " " " ØØ8Ø68

1° operando R1 =

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando R2 =

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ	Ø11Ø1ØØØ

risultato in R1 =

1ØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ	Ø11Ø1ØØØ

Descrizione Funzionale

Carica un registro con il contenuto di un altro registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	LR	R_1, R_2	RR	18

Azione

Il contenuto del registro R2 è caricato nel registro R1.

C.C.: non gestito

Esempio

LR 4,2

dove:

1° operando reg. 4 il cui valore esadecimale è Ø1Ø8BA

2° operando reg. 2 " " " " " ØØ8Ø68

risultato reg. 4 " " " " " ØØ8Ø68

1° operando R1 =

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando R2 =

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ	Ø11Ø1ØØØ

risultato in R1 =

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ	Ø11Ø1ØØØ

Descrizione Funzionale

Carica, esaminandola, il contenuto di una parola di memoria in un registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	LT	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	52

Azione

Il registro R1 è caricato con il contenuto della parola di memoria indirizzata dal secondo operando. Il segno ed il valore assoluto di tale contenuto determinano lo stato del Codice Condizione.

C.C.: Ø il contenuto del secondo operando è = Ø
1 " " " " " " < Ø
2 " " " " " " > Ø

Descrizione Funzionale

Carica un registro con il contenuto di un altro registro e lo esamina.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	LTR	R_1, R_2	RR	12

Azione

Il registro R1 è caricato con il contenuto del registro R2. Il segno ed il valore assoluto di tale contenuto determinano lo stato del Codice Condizione. Se R1 = R2 il contenuto di R1 resta immutato, ma viene gestito il Codice Condizione.

C.C.: Ø il contenuto del secondo operando è = Ø
1 " " " " " " < Ø
2 " " " " " " > Ø

Descrizione Funzionale

Moltiplica il contenuto di una mezza parola per il contenuto di un registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	MLH	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	83

Azione

Il prodotto avviene fra il contenuto della mezza parola allineata indirizzata dal secondo operando, ed il contenuto del registro R1. Il risultato viene messo nel registro R1. Un eventuale overflow non viene segnalato. Il bit più pesante di ciascuno dei due operandi, non viene interpretato come bit di segno. Se uno dei due operandi, o entrambi, sono uguali a zero, si ottiene come risultato uno zero positivo.

C.C.: non gestito

Esempio

MLH 5,60 (,6)

dove:

1° operando reg. 5 il cui valore esadecimale è Ø1Ø8BA

2° operando 6Ø(,6) " " " " " ØØØØØ3

risultato reg. 5 " " " " " Ø31A2E

1° operando R1 =

ØØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando = mezza parola

ØØØØØØØØ	ØØØØØØ11

risultato in R1 =

ØØØØØØØØ	ØØØØØØ11	ØØØ11Ø1Ø	ØØ1Ø111Ø

Descrizione Funzionale

Moltiplica in modo logico il contenuto di un registro per il contenuto di un altro registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	MLR	R_1, R_2	RR	26

Azione

Il prodotto avviene fra i contenuti dei registri R1 e R2. Il risultato viene posto in R1. Un'eventuale overflow non viene segnalato. Se uno dei due operandi, o entrambi, sono uguali a zero, si ottiene come risultato uno zero positivo. Se R1 = R2 il risultato della operazione è il quadrato del contenuto del registro.

C.C.: non gestito

Esempio

MLR 4,1

dove:

1° operando reg. 4 il cui valore esadecimale è Ø1Ø8BA

2. operando reg. 1 " " " " " ØØØØØ3

risultato reg. 4 " " " " " Ø31A2E

1° operando R1 =

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando R2 =

ØØØØØØØØ	ØØØØØØØØ	ØØØØØØØØ	ØØØØØØ11

risultato in R1 =

ØØØØØØØØ	ØØØØØØ11	ØØØ11Ø1Ø	ØØ1Ø111Ø

Descrizione Funzionale

Trasferisce il contenuto di una zona di memoria in un'altra zona di memoria.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	MVC	D_1 (L,B_1), D_2(B_2) S_1 (L), S_2 S_1, S_2 D_1 (, B_1), D_2 (, B_2)	SS	D2

Azione

Un numero di byte pari alla lunghezza L viene mosso dal campo specificato come secondo operando nel campo specificato come primo operando. L è la lunghezza del campo di arrivo. L'istruzione sposta byte per byte, e procede da sinistra verso destra. Il massimo numero di byte che può essere trasferito è 256.

C.C.: non gestito

Descrizione Funzionale

Trasferisce un campo di memoria in un altro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	MVCR	D_1 (B_1), D_2 (B_2) S_1, S_2	SS	F2

Azione

L'istruzione muove il campo indirizzato dal secondo operando nel campo indirizzato dal primo operando. Il numero di byte da trasferire è indicato nel registro Ø. Se il contenuto del registro Ø = Ø, l'operazione non ha nessun effetto. Il movimento tra i campi avviene in modo da garantire il salvataggio delle informazioni contenute nel campo di partenza.

C.C.: non gestito

Descrizione Funzionale

Trasferisce un byte dell'istruzione in una posizione di memoria.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	MVI	D_1 (B_1), I_2 S_1, I_2	SI	92

Azione

Un dato immediato I2 (1 byte), viene trasferito allo indirizzo di memoria specificato dal primo operando.

C.C.: non gestito

Descrizione Funzionale

Trasferisce dal secondo al primo operando solo i semibyte di destra (campi numerici).

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	MVN	D_1 (L, B_1), D_2 (B_2) S_1 (L), S_2 S_1, S_2 D_1 (, B_1), D_2 (, B_2)	SS	D1

Azione

Per ogni byte del campo indirizzato dal secondo operando, il semibyte di destra sostituisce il rispettivo semibyte dei byte che compongono il campo indirizzato dal primo operando. La lunghezza L che regola il trasferimento è quella del primo operando e può variare da 1 a 256 byte. I semibyte di sinistra del primo operando restano invariati.

C.C.: non gestito

Descrizione Funzionale

Il campo di memoria indirizzato dal secondo operando è trasferito a sinistra degli ultimi 4 bit di destra del campo di memoria indirizzato dal primo operando.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	MVO	D_1 (L_1, B_1), D_2 (L_2, B_2) S_1 (L_1), S_2 (L_2) S_1, S_2 D_1 (, B_1), D_2 (, B_2)	S$	F1

Azione

Il campo di memoria indirizzato dal secondo operando, viene trasferito con uno spostamento a sinistra di un semibyte nel campo indirizzato dal primo operando. Il trasferimento ha luogo da destra a sinistra e per la lunghezza massima di 16 byte. Se L1 < L2 si ha un troncamento. Se L1 > L2 l'eccedenza viene riempita con zeri binari.

C.C.: non gestito

Descrizione Funzionale

Trasferisce i semibyte di sinistra (campi zona) di un campo di memoria in un altro campo.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	MVZ	D_1 (L,B_1),D_2 (B_2) S_1 (L), S_2 S_1, S_2 D_1 (, B_1), D_2 (, B_2)	SS	D3

Azione

Per ogni byte del campo indirizzato dal secondo operando, il semibyte di sinistra sostituisce il rispettivo semibyte dei byte che compongono il campo indirizzato dal primo operando. La lunghezza L che regola il trasferimento è quella del primo operando è può variare da 1 a 256 byte. I semibyte di destra del primo operando restano invariati.

C.C.: non gestito

N

Descrizione Funzionale

Operazione di AND tra i contenuti di un registro e di una parola.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	N	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	54

Azione

L'AND fra il contenuto del registro R1 e della parola indirizzata dal secondo operando, sostituisce il contenuto del registro R1. L'operazione segue bit per bit lo schema dell'operazione di AND sotto riportata:

AND	Ø	1
Ø	Ø	Ø
1	Ø	1

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " ≠ Ø

Esempio

N 2,50 (,10)

dove:

1° operando reg. 2 il cui valore esadecimale è Ø1Ø8BA

2° operando 5Ø(,1Ø) " " " " " ØØØØØ3

risultato reg. 2 " " " " " Ø31A2E

1° operando R1 =

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando = parola

ØØØØØØØØ	ØØØØØØØØ	ØØØØØØØØ	ØØØØØØ11

risultato in R1 =

ØØØØØØØØ	ØØØØØØ11	ØØØ11Ø1Ø	ØØ1Ø111Ø

NC

Descrizione Funzionale — AND fra i contenuti di due campi di memoria.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	NC	D_1 (L,B_1), D_2 (B_2) S_1 (L), S_2 S_1, S_2 D_1 (, B_1), D_2 (, B_2)	SS	D4

Azione

L'AND tra il contenuto del campo di memoria primo operando, di lunghezza L byte, e del campo di memoria secondo operando, pure di L byte, sostituisce il contenuto del primo operando. La lunghezza L è quella associata implicitamente o esplicitamente al campo contenuto nel primo operando. Qualunque configurazione di bit è valida. Le regole seguite sono quelle dell'operazione logica di AND (vedi istruzione N).

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " ≠ Ø

Esempio

NC 16 (4,11), 50 (4,13)

dove:

1° operando 16(4,11) il cui valore esadecimale è Ø1Ø8BA

2° operando 5Ø(4,13) " " " " " ØØØØØ3

risultato 16(4,11) " " " " " Ø31A2E

1° operando = campo di memoria

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando = campo di memoria

ØØØØØØØØ	ØØØØØØØØ	ØØØØØØØØ	ØØØØØØ11

risultato in 1° operando

ØØØØØØØØ	ØØØØØØ11	ØØØ11Ø1Ø	ØØ1Ø111Ø

Descrizione Funzionale

AND fra il byte contenuto nella istruzione e un byte di memoria.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	NI	D_1 (B_1), I_2 S_1, I_2	SI	94

Azione

L'AND fra l'immediato I2 e il byte di memoria, sostituisce il contenuto del primo operando. Qualunque configurazione di bit è valida. Le regole seguite sono quelle della operazione logica di AND (vedi istruzione N).

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " ≠ Ø

Esempio

NI 30 (5), X' 05'

dove:

1° operando 3Ø(5) il cui valore esadecimale è 1F

2° operando x'Ø5' " " " " " Ø5

risultato 3Ø(5) " " " " " 9B

1° operando = byte di memoria

| ØØØ11111 |

2° operando I2 =

| ØØØØØ1Ø1 |

risultato in 1° operando

| 1ØØ11Ø11 |

NOP

Descrizione Funzionale — Nessuna esecuzione di operazioni.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	NOP	$M_1, D_2 (X_2, B_2)$ $M_1, D_2 (, B_2)$ $M_1, S_2 (X_2)$ M_1, S_2	RX	470

Azione

La maschera presente nella istruzione deve essere sempre posta a zero, quindi il Condition Code non viene testato e l'istruzione non esegue alcun salto. Viene quindi eseguita la prossima istruzione in sequenza. Il codice operativo è espresso in codice esadecimale lungo 1 byte e mezzo. Questa istruzione permette di creare spazio per l'introduzione successiva di istruzioni (vedere istruzione BC).

C.C.: non gestito

Descrizione Funzionale — Nessuna esecuzione di operazioni.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	NOPR	M_1, R_2	RR	070

Azione

La maschera presente nella istruzione deve essere sempre posta a zero, quindi il Codice Condizione non viene testato e l'istruzione non esegue alcun salto. Viene quindi eseguita la prossima istruzione in sequenza. Per quanto riguarda codice operativo e maschera vedere istruzione NOP. Questa istruzione permette di creare spazio per l'introduzione successiva di istruzioni (vedere istruzione BCR).

C.C.: non gestito

NR

Descrizione Funzionale — AND fra il contenuto di due registri.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	NR	R_1, R_2	RR	14

Azione

L'AND fra i contenuti dei registri R1 e R2 sostituisce il contenuto del registro R1. Qualunaue configurazione di bit è valida. Le regole seguite sono quelle della operazione logica di AND (vedi istruzione N).

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " ≠ Ø

Esempio

NR 5,4

dove:

1° operando reg. 5 il cui valore esadecimale è Ø1Ø8BA

2° operando reg. 4 " " " " " ØØØØØ3

risultato reg 5 " " " " " Ø31A2E

1° operando R1 =

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando R2 =

ØØØØØØØØ	ØØØØØØØØ	ØØØØØØØØ	ØØØØØØ11

risultato in R1 =

ØØØØØØØØ	ØØØØØØ11	ØØØ11Ø1Ø	ØØ1Ø1111Ø

Descrizione Funzionale — OR fra i contenuti di un registro e di una parola.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	O	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	56

Azione

L'OR fra i contenuti del registro R1 e della parola allineata contenuta nel secondo operando sostituisce il contenuto di R1. Qualunque configurazione di bit è valida. L'operazione segue bit per bit lo schema dell'operazione logica di OR sotto riportata:

OR	Ø	1
Ø	Ø	1
1	1	1

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " ≠ Ø

Esempio :

O 13,100 (4,6)

dove:

1° operando reg. 13 il cui valore esadecimale è Ø1Ø8BA

2° operando 1ØØ(4,6) " " " " " ØØ8Ø68

risultato reg. 13 " " " " " Ø188FA

1° operando R1 =

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando = parola

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ	Ø11Ø1ØØØ

risultato in R1 =

1ØØØØØØØ	ØØØØØØØ1	1ØØØ1ØØØ	11111Ø1Ø

Descrizione Funzionale — OR fra i contenuti di due campi di memoria.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	OC	D_1 (L,B_1), D_2 (B_2) S_1 (L), S_2 S_1, S_2 D_1 (, B_1), D_2 (, B_2)	SS	D6

Azione

L'OR fra il contenuto del campo di memoria del primo operando, di lunghezza L byte, e del campo di memoria del secondo operando, pure lungo L byte, sostituisce il contenuto del primo operando. La lunghezza L è quella associata implicitamente o esplicitamente al primo operando. qualunque configurazione di bit è valida. L'operazione segue bit per bit lo schema della operazione logica di OR (vedi istruzione O).

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " ≠ Ø

Esempio

OC 50 (4,7), 26 (4,11)

dove:

1° operando 5Ø(4,7) il cui valore esadecimale è Ø1Ø8BA

2° operando 26(4,11) " " " " " ØØ8Ø68

risultato 5Ø(4,7) " " " " " Ø188FA

1° operando = campo di memoria

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando = campo di memoria

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ	Ø11Ø1ØØØ

risultato in 1° operando

1ØØØØØØØ	ØØØØØØØ1	1ØØØ1ØØØ	11111Ø1Ø

Descrizione Funzionale

OR fra un byte contenuto nell'istruzione e un byte di memoria.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	OI	D_1 (B_1), I_2 S_1, I_2	SI	96

Azione

L'OR fra l'immediato I2 e il byte di memoria indirizzato dal primo operando, sostituisce il contenuto del primo operando. Qualunque configurazione di bit è valida. L'operazione segue bit per bit lo schema della operazione logica di OR (vedi istruzione O).

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " ≠ Ø

Esempio

OI 10 (10), X'68'

dove:

1° operando 1Ø(1Ø) il cui valore esadecimale è BA

2° operando x'68' " " " " " 68

risultato 1Ø(1Ø) " " " " " FA

1° operando = byte di memoria

1Ø111Ø1Ø

2° operando I2 =

Ø11Ø1ØØØ

risultato in 1° operando

11111Ø1Ø

Descrizione Funzionale — OR del contenuto di due registri.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	OR	R_1, R_2	RR	16

Azione

L'OR fra il contenuto dei registri R1 e R2 sostituisce il contenuto di R1. Qualunque configurazione di bit è valida. L'operazione segue bit per bit lo schema della operazione logica di OR (vedi istruzione O).

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " ≠ Ø

Esempio

OR 5,4

dove:

1° operando reg. 5 il cui valore esadecimale è Ø1Ø8BA

2° operando reg. 4 " " " " " ØØ8Ø68

risultato reg. 5 " " " " " Ø188FA

1° operando R1 =

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando R2 =

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ	Ø11Ø1ØØØ

risultato in R1 =

1ØØØØØØØ	ØØØØØØØ1	1ØØØ1ØØØ	11111Ø1Ø

RETS

Descrizione Funzionale

Dealloca la lista degli argomenti e salta al programma chiamato.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	RETS	I	RR	20

Azione

Dealloca la lista degli argomenti dello stack e salta all'istruzione di indirizzo pari a quello dell'istruzione di chiamata, memorizzato sullo stack, incrementato del valore dell'immediato I.

Descrizione Funzionale

Sottrae il contenuto di una parola al contenuto di un registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	S	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	5B

Azione

Il contenuto della parola indirizzata dal secondo operando è sottratto al contenuto del registro R1, ed il risultato è posto in R1. L'operazione è svolta sommando il complemento a 2 della parola al registro. Tutti i bit dei due operandi (segni compresi) partecipano all'operazione. L'overflow si verifica quando i riporti del segno e del bit di ordine più alto sono discordanti.

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " < Ø
2 " " " " " > Ø
3 se si verifica un'overflow.

Esempio

S 12,64 (4,6)

dove:

1° operando reg. 12 il cui valore esadecimale è Ø1Ø8BA

2° operando 64(4,6) " " " " " ØØ8Ø68

risultato reg. 12 " " " " " ØØ8852

1° operando R1 =

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ 1Ø111Ø1Ø

2° operando = parola

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ Ø11Ø1ØØØ

risultato in R1 =

1ØØØØØØØ	ØØØØØØØØ	1ØØØ1ØØØ Ø1Ø1ØØ1Ø

Descrizione Funzionale

Ricerca sequenziale di un elemento sul sottocampo di una tabella.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	SES	D_1 (L_1, B_1), D_2 (L_2, B_2) S_1 (L_1), S_2 (L_2) S_1, S_2 D_1 (, B_1), D_2 (, B_2)	SS	F7

Azione

Viene ricercata l'uguaglianza di un sottocampo dell'area indirizzata dal primo operando e del campo di memoria indirizzato dal secondo operando. Il primo campo si intende logicamente organizzato in gruppi di L1 byte ciascuno. Il confronto avviene, successivamente, per un numero L2 di byte, fra il campo di memoria secondo operando e il sottocampo di ognuno dei gruppi di L1 byte distante un numero di byte pari al contenuto del registro generale 2 dall'inizio del gruppo stesso. Del registro 2 sono presi in considerazione solo i quattro bit meno significativi. La ricerca procede per un numero di volte pari al contenuto del registro generale Ø.

Se l'esito della ricerca è positivo, viene caricato nel registro generale 1 l'indirizzo del campo il cui sottocampo ha verificato l'uguaglianza. Nel registro generale 2 il numero ordinale (Ø÷N) dell'elemento nella tabella.

Se l'esito è negativo, si carica nel registro generale 1 l'indirizzo del byte successivo all'ultimo elemento, e nel registro generale 2 il numero totale di elementi (N + 1). Se il registro generale Ø = Ø si pone il condition code = 3 e si carica nel registro

generale 1 l'indirizzo della tabella, e nel registro generale 2 il valore Ø.

C.C.: Ø se la ricerca ha esito positivo
1 " " " " " "
2 " " " " " "
3 se la ricerca ha esito negativo

Descrizione Funzionale

Ricerca sequenziale di un elemento sul sottocampo mascherato di una tabella.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	SESM	D_1 (L_1, B_1), D_2 (L_2, B_2) S_1 (L_1), S_2 (L_2) S_1, S_2 D_1 (, B_1), D_2 (, B_2)	SS	F0

Azione

Viene fatto il confronto fra il risultato dell'AND di un sottocampo della tabella indirizzata dal primo operando con gli L2 byte meno significativi del registro R1, e il campo di memoria indirizzato dal secondo operando. La ricerca è effettuata in modo sequenziale a partire dall'elemento indirizzato e per indirizzi crescenti di memoria. L1 è la lunghezza di un elemento di tabella. L2 è la lunghezza del secondo operando e quindi anche del sottocampo da confrontare. Se L2 > 4 si assume L2 = 4. Il registro generale Ø contiene il numero totale degli elementi della tabella da considerare (solo i 2 byte meno pesanti sono ritenuti significativi). Il registro generale 2 contiene lo spiazzamento del sottocampo nell'elemento. Si considerano solo i 4 bit meno pesanti del registro generale 2.

Se l'esito della ricerca è positivo, si carica nel registro generale 1 l'indirizzo dell'elemento il cui sottocampo ha causato il confronto, e nel registro generale 2 il numero ordinale (Ø ÷ N) dell'elemento nella tabella.

Se l'esito è negativo, si carica nel registro generale

1 l'indirizzo del byte successivo all'ultimo elemento, e nel registro generale 2 il numero totale di elementi (N + 1). Se il registro generale Ø = Ø si pone il condition code a 3 e si carica nel registro generale 1 l'indirizzo della tabella e nel registro generale 2 il valore Ø.

C.C.: Ø se la ricerca ha esito positivo
1 " " " " " "
2 " " " " " "
3 se la ricerca ha esito negativo

Descrizione Funzionale

Sottrae il contenuto di una mezza parola dal contenuto di un registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	SH	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	4B

Azione

Il contenuto della mezza parola allineata e indirizzata dal secondo operando è sottratto al contenuto del registro R1 ed il risultato è posto in R1. Prima dell'operazione il segno della mezza voce viene espanso per una lunghezza di 16 bit; l'operazione è eseguita sommando il complemento a 2 del valore così ottenuto (32 bit) al registro. Un'overflow si verifica quando i riporti del segno e del bit di ordine più alto sono discordanti.

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " < Ø
2 " " " " " " > Ø
3 se si verifica un'overflow

Esempio

SH 9,52 (,2)

dove:

1° operando reg. 9 il cui valore esadecimale è Ø1Ø8BA

2° operando 52(,2) " " " " " FF8Ø68

risultato reg. 9 " " " " " Ø18852

1° operando R1 =

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ 1Ø111Ø1Ø

2° operando = mezza parola

11111111	11111111	1ØØØØØØØ Ø11Ø1ØØØ

ESPANSIONE

risultato il R1 =

1ØØØØØØØ	ØØØØØØØ1	1ØØØ1ØØØ Ø1Ø1ØØ11

Descrizione Funzionale

Sottrae in modo logico il contenuto di una parola dal contenuto di un registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	SL	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	5F

Azione

Il contenuto della parola allineata indirizzata dal secondo operando è sottratta al contenuto del registro R1 ed il risultato è posto in R1. L'operazione è eseguita sommando il complemento a due del contenuto della voce al registro. Se si verifica un riporto della posizione del segno il condition code viene posto a determinati valori, come specificato in seguito. Si noti che il risultato Ø non può essere ottenuto senza che si verifichi un riporto; pertanto il condition code non sarà mai posto a Ø.

C.C.: Ø se il risultato è ≠ Ø e non c'è riporto
2 se il risultato è = Ø e c'è riporto
3 " " " " ≠ Ø " " "

Esempio

SL 5,17 (,8)

dove:

1° operando reg. 5 il cui valore esadecimale è Ø1Ø8BA

2° operando 17(,8) " " " " " ØØ8Ø68

risultato reg. 5 " " " " " ØØ8852

1° operando R1 =

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ 1Ø111Ø1Ø

2° operando = parola

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ Ø11Ø1ØØØ

risultato in R1 =

1ØØØØØØØ	ØØØØØØØØ	1ØØØ1ØØØ Ø1Ø1ØØ1Ø

SHIFT LEFT ALGEBRAIC

SLA

Descrizione Funzionale

Trasla a sinistra il contenuto di un registro, segno escluso.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	SLA	R_1, I_2	RR	02

Azione

Il contenuto di R1 è traslato a sinistra di tante posizioni, quante specificate dall'immediato I2. Il segno resta immutato. I bit che escono a sinistra dai limiti del registro sono persi, mentre altrettanti Ø entrano da destra. Uno spostamento del contenuto di un registro a sinistra, equivale a moltiplicarlo per 2. Si verifica overflow quando viene perduto per lo shift un bit diverso dal bit del segno.

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " < Ø
2 " " " " " " > Ø
3 se si verifica un'overflow

Descrizione Funzionale — Trasla a sinistra il contenuto di un registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	SLL	R_1, I_2	RR	04

Azione

Il contenuto di R1 è traslato a sinistra di tante posizioni quante specificate dall'immediato I2. Tutti i bit di R1, segno compreso, partecipano all'operazione. I bit che escono a sinistra sono persi, e altrettanti Ø entrano da destra.

C.C.: non gestito

Esempio

SLM 50 (,6), 26 (,7)

dove:

1° operando 5Ø(,6) il cui valore esadecimale è Ø1Ø8BA

2° operando 26(,7) " " " " " ØØ8Ø68

risultato 5Ø(,6) " " " " " ØØ8852

1° operando = campo di memoria

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ 1Ø111Ø1Ø

2° operando = campo di memoria

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ Ø11Ø1ØØØ

risultato in 1° operando

1ØØØØØØØ	ØØØØØØØØ	1ØØØ1ØØØ Ø1Ø1ØØ1Ø

Descrizione Funzionale

Sottrae il contenuto di un campo di memoria ad un altro campo di memoria.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	SLM	D_1 (L_1, B_1), D_2 (L_2, B_2) S_1 (L_1), S_2 (L_2) S_1, S_2 D_1 (, B_1), D_2 (, B_2)	SS	F4

Azione

Il contenuto del campo di memoria indirizzato dal secondo operando è sottratto al contenuto del campo di memoria indirizzato dal primo operando. La sottrazione viene ottenuta sommando il complemento a 2 del secondo operando al primo operando. L'operazione procede da destra a sinistra per il numero di byte indicato in L1. Se il secondo operando è più corto viene esteso "logicamente" con bit Ø.

C.C.: 1 se il risultato è ≠ Ø senza riporto
2 " " " " = Ø con riporto
3 " " " " ≠ Ø con riporto

Descrizione Funzionale

Sottrae in modo logico il contenuto di un registro al contenuto di un altro registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	SLR	R_1, R_2	RR	1F

Azione

Il contenuto di R2 è sottratto al contenuto di R1 e il risultato è posto in R1. L'operazione è eseguita sommando il complemento a 2 del contenuto di R2 al contenuto di R1. Tutti i bit dei due operandi partecipano all'operazione. Se si verifica un riporto dalla posizione del segno, il condition code viene posizionato come descritto in seguito. Si noti che il risultato Ø non può essere ottenuto senza che si verifichi un riporto, pertanto il Codice Condizione non sarà mai impostato a Ø.

C.C.: 1 se il risultato è ≠ Ø e non c'è riporto
2 " " " " = Ø e c'è riporto
3 " " " " ≠ Ø e c'è riporto

Esempio

SLR 4,11

dove:

1° operando reg. 4 il cui valore esadecimale è Ø1Ø8BA

2° operando reg. 11 " " " " " 888Ø68

risultato reg. 4 " " " " " ØØ8852

1° operando R1 =

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ 1Ø111Ø1Ø

2° operando R2 =

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ Ø11Ø1ØØØ

risultato in R1 =

1ØØØØØØØ	ØØØØØØØØ	1ØØØ1ØØØ Ø1Ø1ØØ1Ø

Descrizione Funzionale

Sottrae un immediato al contenuto di un registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	SLRI	R_1, I_2	RR	0C

Azione

Sottrae al contenuto del registro R1 l'immediato I2. L'eventuale riporto viene segnalato dal condition code. La sottrazione avviene con la somma al registro R1 del complemento a 2 di I2.

C.C.: Ø se la differenza senza riporto è = Ø
1 " " " " " " ≠ Ø
2 se la differenza con riporto è = Ø
3 " " " " " " ≠ Ø

Descrizione Funzionale

Sottrae algebricamente il contenuto di un campo di memoria dal contenuto di un altro campo di memoria.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	SM	D_1 (L_1, B_1), D_2 (L_2, B_2) S_1 (L_1), S_2 (L_2) S_1, S_2 D_1 (, B_1), D_2 (, B_2)	SS	F3

Azione

Il campo di memoria indirizzato dal secondo operando è sottratto al campo di memoria indirizzato dal primo operando; il risultato è posto nel primo operando. Nell'operazione il bit più pesante del byte più pesante di ciascun campo, è considerato come bit di segno. L'operazione viene eseguita espandendo logicamente con il bit del segno l'operando più corto, e sommando al primo operando il complemento a 2 del secondo operando. Si verifica overflow:

- se i riporti del segno e del bit di ordine più alto sono discordanti

- se L1 < L2 e gli L2 - L1 byte più pesanti del risultato non sono costituiti da bit uguali al bit più pesante del byte L1 del risultato.

C.C.: Ø se il risultato è = Ø
1 " " " " < Ø
2 " " " " > Ø
3 se si verifica overflow

Esempio

SM 48 (2,5), 64 (1,7)

dove:

1° operando 48(2,5) il cui valore esadecimale è Ø8BA

2° operando 64(1,7) " " " " " ØØ68

risultato 48(2,5) " " " " " Ø852

1° operando = campo di memoria

ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando = campo di memoria

ØØØØØØØØ	Ø11Ø1ØØØ

ESPANSIONE

risultato in 1° operando

ØØØØ1ØØØ	Ø1Ø1ØØ1Ø

Descrizione Funzionale

Sottrae il contenuto di un registro al contenuto di un altro registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	SR	R_1, R_2	RR	1B

Azione

Il contenuto di R2 è sottratto al contenuto di R1, e il risultato è messo in R1. L'operazione è eseguita sommando il complemento a 2 di R2 a R1. Ponendo R1 = R2 si ha come risultato l'azzeramento del registro. Un'overflow si verifica quando i riporti del segno e del bit di ordine più alto sono discordanti.

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " < Ø
2 " " " " " " > Ø
3 se si verifica un'overflow

Esempio

SR 8,3

dove:

1° operando reg. 8 il cui valore esadecimale è FFFFFF

2° operando reg. 3 " " " " " FFFFFF

risultato reg. 8 " " " " " ØØØØØØ

1° operando R1 =

11111111	11111111	11111111	11111111

2° operando R2 =

11111111	11111111	11111111	11111111

risultato in R1 =

ØØØØØØØØ	ØØØØØØØØ	ØØØØØØØØ	ØØØØØØØØ

Descrizione Funzionale

Trasla a destra il contenuto di un registro, segno escluso.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	SRA	R_1, I_2	RR	03

Azione

Il contenuto di R1 è traslato a destra di tante posizioni quante sono specificate dall'immediato I2. Il segno resta immutato. I bit che escono a destra sono persi, mentre quelli che entrano da sinistra sono uguali al segno.

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " Ø
2 " " " " " " Ø

Descrizione Funzionale

Trasla a destra il contenuto di un registro , segno compreso.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	SRL	R_1, I_2	RR	08

Azione

Il contenuto di R1 è traslato a destra di tante posizioni quante specificate dall'immediato I2. Tutti i bit segno compreso partecipano all'operazione. I bit che escono da destra sono persi; da sinistra entrano degli Ø.

C.C.: non gestito

Descrizione Funzionale

Memorizza il contenuto di un registro in una parola.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	ST	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	50

Azione

Il contenuto del registro R1 è memorizzato nel campo di memoria specificato come secondo operando. Il campo di memoria indirizzato dal secondo operando deve essere una parola.

C.C.: non gestito.

Descrizione Funzionale

Memorizza in un byte di memoria gli ultimi 8 bit di destra di un registro.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	STC	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	42

Azione

I bit 24 ÷ 31 del registro R1 sono memorizzati all'indirizzo espresso dal secondo operando.

C.C.: non gestito

STH

Descrizione Funzionale

Memorizza il contenuto dei due byte di destra di un registro in una mezza parola.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	STH	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	40

Azione

Il contenuto della metà di destra del registro R1 è memorizzato alla mezza parola indirizzata dal secondo operando. La metà sinistra del registro non partecipa all'operazione.

C.C.: non gestito

Descrizione Funzionale

Memorizza il contenuto di N registri adiacenti in N parole consecutive.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	STM	$R_1, R_2, D_3 (B_3)$ R_1, R_2, S_3	RS	90

Azione

Il contenuto dei registri compresi tra R1...R2 è memorizzato in altrettante parole consecutive, a partire dall'indirizzo espresso dal secondo operando. Se R1 = R2 il solo registro R1 è memorizzato. Se R1 > R2 il trasferimento ha comunque luogo, tenuto conto che il registro RØ è considerato consecutivo al registro R15.

C.C.: non gestito.

Descrizione Funzionale

Esamina un byte di memoria usando una maschera inserita nell'istruzione.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	TM	D_1 (B_1), I_2 S_1, I_2	SI	91

Azione

L'immediato specificato nell'istruzione è la maschera. Per ogni bit = 1 ad esso appartenente, viene controllato il bit corrispondente del byte di memoria indirizzato dal primo operando.

C.C.: Ø se tutti i bit controllati sono = Ø oppure
se tutti i bit della maschera sono = Ø
1 se tutti i bit controllati sono misti
3 se tutti i bit controllati sono = 1

Descrizione Funzionale Ricerca tabellare.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	TR	D_1 (L, B_1), D_2 (B_2) S_1 (L), S_2 S_1, S_2 D_1 (, B_1), D_2 (, B_2)	SS	DC

Azione

I byte contenuti nel campo indirizzato dal 1° operando sono considerati come una lista di argomenti; quelli del campo indirizzato dal 2°operando come una lista di funzioni. Vengono presi in considerazione uno alla volta i byte la lista degli argomenti; il contenuto di ognuno di essi preso come valore intero non segnato viene sommato in modo aritmetico all'indirizzo del secondo operando. Il nuovo indirizzo così ottenuto corrisponde ad un byte della lista delle funzioni. Il contenuto del byte funzione sostituisce quello del byte argomento. L'operazione continua fino all'esaurimento del campo argomenti.

C.C.: non gestito

Descrizione Funzionale

Ricerca tabellare con test della funzione.

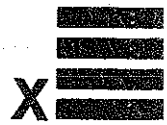
Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	TRT	D_1 (L, B_1), D_2 (B_2) S_1 (L), S_2 S_1, S_2 D_1 (, B_1), D_2 (, B_2)	SS	D0

Azione

I byte contenuti nel campo indirizzato dal primo operando sono considerati come una lista di argomenti; quelli contenuti nel campo indirizzato dal secondo operando sono considerati come una lista di funzioni. Partendo da sinistra e per un numero di byte pari ad L (1 ÷ 256) vengono presi in considerazione uno alla volta i byte formanti la lista degli argomenti. Il contenuto di ognuno di essi viene sommato all'indirizzo del secondo operando; il nuovo indirizzo così ottenuto corrisponde ad un byte della lista delle funzioni. Se l'AND di quest'ultimo byte con il byte meno pesante del registro generale Ø è uguale a Ø, viene preso in esame il successivo argomento; se invece è diverso da Ø, lo stesso byte viene inserito nelle 8 posizioni binarie di destra del registro generale 2 (il resto del registro resta immutato), lo indirizzo dell'argomento corrispondente, viene inserito nel registro generale 1 (il resto del registro resta immutato) e l'operatore si arresta. I byte formanti la lista degli argomenti restano immutati.

C.C.: Ø se tutti i byte controllati dalla lista delle funzioni sono = Ø
1 se almeno un byte controllato dalla lista delle funzioni è ≠ Ø
2 se l'ultimo byte controllato selezionato nella lista delle funzioni è ≠ Ø



Descrizione Funzionale

OR esclusivo fra i contenuti di un registro e di una parola.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	X	$R_1, D_2 (X_2, B_2)$ $R_1, D_2 (, B_2)$ $R_1, S_2 (X_2)$ R_1, S_2	RX	57

Azione

L'OR esclusivo fra i contenuti del registro R1 e della parola indirizzata dal secondo operando, sostituisce il contenuto di R1. Qualunque configurazione di bit è valida. L'operazione segue bit per bit lo schema dell'OR esclusivo sotto riportata:

XOR	Ø	1
Ø	Ø	1
1	1	Ø

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " ≠ Ø

Esempio

X 12,37 (4,6)

dove:

1° operando reg. 12 il cui valore esadecimale è Ø1Ø8BA

2° operando 37(4,6) " " " " " ØØ8Ø68

risultato reg. 12 " " " " " Ø188D2

1° operando R1 =

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando = parola

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ	Ø11Ø1ØØØ

risultato in R1 =

1ØØØØØØØ	ØØØØØØØ1	1ØØØ1ØØØ	11Ø1ØØ1Ø

Descrizione Funzionale — OR esclusivo fra il contenuto di due campi di memoria.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	XC	D_1 (L,B_1), D_2 (B_2) S_1 (L), S_2 S_1, S_2 D_1 (, B_1), D_2 (, B_2)	SS	D7

Azione

L'OR esclusivo del contenuto del campo di memoria indirizzato dal primo operando, di lunghezza L byte, e del campo di memoria indirizzato dal secondo operando, pure lungo L byte, sostituisce il contenuto del primo operando. La lunghezza L è quella associata implicitamente o esplicitamente al campo contenuto nel primo operando. Qualunque configurazione di bit è valida. L'operazione segue bit per bit lo schema dell'operazione di OR exclusivo (vidi istruzione X).

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " ≠ Ø

Esempio

XC 97 (,5), 84 (,9)

dove:

1° operando 97(,5) il cui valore esadecimale è Ø1Ø8BA

2° operando 84(,9) " " " " " ØØ8Ø68

risultato 97(,5) " " " " " Ø188D2

1° operando = campo di memoria

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando = campo di memoria

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ	Ø11Ø1ØØØ

risultato in 1° operando =

1ØØØØØØØ	ØØØØØØØ1	1ØØØ1ØØØ	11Ø1ØØ1Ø

Descrizione Funzionale — OR esclusivo di un byte di memoria con un immediato.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	XI	D_1 (B_1), I_2 S_1, I_2	SI	97

Azione

L'OR esclusivo dell'immediato I2 e del byte di memoria indirizzato dal primo operando, sostituisce il contenuto del primo operando. Qualunque configurazione di bit è valida. L'operazione segue bit per bit lo schema dell'operazione di OR esclusivo (vedi istruzione X).

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " ≠ Ø

Esempio

XI 43 (5), X'6E'

dove:

1° operando 43(5) il cui valore esadecimale è BB

2° operando x'6E' " " " " " 6E

risultato 43(5) " " " " " D5

1° operando = byte di memoria

1Ø111Ø11

2° operando I2 =

Ø11Ø111Ø

risultato in 1° operando

11Ø1Ø1Ø1

Descrizione Funzionale — OR esclusivo dei contenuti di due registri.

Formato

nome	operazione	operando	tipo di formato di memoria	codice operativo
[simbolo]	XR	R_1, R_2	RR	17

Azione

L'OR esclusivo dei contenuti dei registri R1 e R2 sostituisce il contenuto di R1. Qualunque configurazione di bit è valida. L'operazione segue bit per bit lo schema della operazione di OR esclusivo (vedi istruzione X).

C.C.: Ø se il risultato dell'operazione è = Ø
1 " " " " " " ≠ Ø

Esempio

XR 12,2

dove:

1° operando reg. 12 il cui valore esadecimale è Ø1Ø8BA

2° operando reg. 2 " " " " " ØØ8Ø68

risultato reg. 12 " " " " " Ø188D2

1° operando R1 =

1ØØØØØØØ	ØØØØØØØ1	ØØØØ1ØØØ	1Ø111Ø1Ø

2° operando R2 =

ØØØØØØØØ	ØØØØØØØØ	1ØØØØØØØ	Ø11Ø1ØØØ

risultato in R1 =

1ØØØØØØØ	ØØØØØØØ1	1ØØØ1ØØØ	11Ø1ØØ1Ø

Printed in Italy